Transform Domain Filtering in Incremental and Diffusion Strategies over Distributed Networks

A Thesis submitted in partial fulfillment of the Requirements for the degree of

Master of Technology

In

SIGNAL AND IMAGE PROCESSING

by

Sumit Kumar

Roll No: 212EC6189

Department of Electronics and Communication Engineering

National Institute of Technology Rourkela

Rourkela, Odisha, 769008

May 2014

Transform Domain Filtering in Incremental and Diffusion Strategies over Distributed Networks

A Thesis submitted in partial fulfillment of the Requirements for the degree of

Master of Technology

In

SIGNAL AND IMAGE PROCESSING

by

Sumit Kumar

Roll No: 212EC6189

Under the Guidance of

Prof. U. K. Sahoo

Department of Electronics and Communication Engineering

National Institute of Technology Rourkela

Rourkela, Odisha, 769008

May 2014

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA

ROURKELA- 769008, ODISHA, INDIA

CERTIFICATE

This is to certify that the work in this thesis entitled "**Transform Domain Filtering in Incremental and Diffusion Strategies over Distributed Networks"** by Mr. **SUMIT KUMAR** is a record of an original research work carried out by his during 2013-2014 under my supervision and guidance in partial fulfilment of the requirement for the award of the degree of Master of Technology in Electronics and Communication Engineering (Signal and Image Processing), National Institute of Technology, Rourkela. Neither this thesis nor any part of it, to the best of my knowledge, has been submitted for any degree or diploma elsewhere.

Place: NIT Rourkela

Date: 30th May 2014

Prof. U. K. Sahoo

Assistant Professor

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA

ROURKELA- 769008, ODISHA, INDIA

Declaration

I certify that

a) The work comprised in the thesis is original and is done by myself under the supervision of my supervisor.

b) The work has not been submitted to any other institute for any degree or diploma.

c) I have followed the guidelines provided by the Institute in writing the thesis.

d) Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them in the text of the thesis and giving their details in the references.

e) Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Sumit Kumar

212EC6189

Acknowledgements

The work posed in this thesis is by far the most substantial attainment in my life and it would be unimaginable without people who affirmed me and believed in me. First and foremost I evince my profound reverence and deep regards to my guide Prof. U. K. Sahoo for exemplary guidance, supervising and constant encouragement throughout the course of this thesis. A gentleman embodied, in true form and spirit, I consider it to my good fortune to have consociated with him.

I would like to evince a deep sense of gratitude to estimable Prof. S. Meher, Head of the Department of Electronics and Communication Engineering for providing us with best facilities and his timely suggestions.

My special thanks to Prof. L.P. Roy, Prof. A. K. Sahoo of Department of Electronics and Communication Engineering for their constant inspiration and encouragement during my research. I want to thank all other faculty members of Department of Electronics and Communication Engineering for their constant support and encouragement during my research. My special thanks to Ph.D. scholars Sananda Kumar for their help, cooperation and encouragement. I would like to thank all my friends who made my journey at NIT Rourkela an indelible and gratifying experience.

Finally, my heartfelt gratitude towards my family for their tireless love and support throughout my life. They taught me the value of hard work by their own life example. They gave me tremendous support during my stay in NIT Rourkela.

Sumit Kumar

List of contents

List of Figures

Abstract

We analyse incremental and diffusion co-operative schemes in which nodes share information to some neighbour nodes in order to estimate desired parameter of interest locally in the presence of noise. Each node works as an adaptive filter and having its own learning ability. In incremental co-operative fashion a node takes information from previous node and after local estimation the information is sent to next node whereas in diffusion the input is taken from various nodes so that after each iteration the behaviour of distributed network is observed. We employ LMS structure for updating the observations.

The convergence performance and computational complexity of LMS-filter is very important consideration for the point of view of speed boost and cost reduction. The convergence performance of a filter depends on eigenvalue spread of covariance matrix of input data or in other words inversely proportional to the eigenvalue spread of the input data. If input data is de-correlated the eigenvalue spread is less and if input data is correlated the eigenvalue spread is more. Transform domain filter has data de-correlation properties of transforms like DCT & DFT. The data de-correlation by the unitary transforms is depends on the orthogonal property of individual transform. Hence we get improved convergence performance by applying transform domain to input data followed by power normalization of input data. If the input data is fully de-correlated the covariance matrix of input data is proportional to the identity matrix. Similarly when a FIR filter has long length of filter coefficients, then the computation cost becomes very high. Which results to time consuming for real time applications. Block adaptive filter makes processing block-by-block rather sample-by-sample in order to reduce the cost factor. Block Adaptive filters of various type are employed to reduce the computational cost of filter. Block adaptive filters are employed via DFT, DCT, DHT and Overlap-Add DFT methods, which are capable to lead better convergence performance as well as better cost reduction.

We achieved good convergence performance and less computational cost. The response in simulation seems very good and mean square error (MSE) is plotted for various methods.

Key-words: - Diffusion, Distributed Network, DFT, DCT, DHT, Overlap-Add DFT, Block Adaptive Filter.

CHAPTER 1

INTRODUCTION

We have a distributed network in which many nodes are distributed spatially. The work of nodes is to observe the temporal data which is coming from various spatial resources of vary different profiles in statistical sense. The nodes have to observe the desired parameter of interest which is coming from one or more spatially resources. The unwanted observations are kept in category of noise. In diffusion co-operative scheme the nodes exchange information with some other neighborhood nodes and update the observations every iteration of time. Each node works as an individual adaptive filter in order to estimate the desired parameter through local observations [1] – [2]. The estimation of desired parameter is done individually by each node and these estimations are locally fused to neighboring connected nodes for their further estimation respectively. Time to time local estimation of each get updated and local fusion by neighbor nodes provides spatial data that make the scheme co-operative in fashion. We can say that estimation at each node is depend on both temporal data and spatial data provided by neighboring nodes [6] – [12]. By the creation of this structure, distributed network is ready to respond in real time situation for different statistical profile of both temporal and spatial data [3] - [4].

Each node works as an adaptive filter and least mean square (LMS) is employed to update the estimation at every iteration. Now the question is the convergence of LMS, which open a totally different section of work. We need lowest settling time or fast convergence in order to work for real time environment. A transform domain approach is employed here to improve the convergence. The basic function of unitary transform like Discrete Fourier Transform (DFT) and Discrete Cosine Transform (DCT) to de-correlate the input data. First the input data is processed by unitary transforms and then power normalization of input data takes place [5]. The convergence performance of a filter depends on eigenvalue spread of covariance matrix of input data or in other words inversely proportional to the eigenvalue spread of the input data. If input data is de-correlated the eigenvalue spread of covariance matrix of input data is less, and if input data is correlated the eigenvalue spread of covariance matrix of input data is more. Hence it is far better to apply transform domain processing. We get improved convergence

performance. If the input data is fully de-correlated the covariance matrix of input data is proportional to the identity matrix.

Another important section of work is computational complexity or computational cost of entire processing, because in distributed network a large number of nodes are employed and each node estimates the desired parameter of interest locally at every iteration of time. The mathematical calculation during local estimation play an important role in speed of estimation. If computational cost is high, our distributed network is less suitable for real time applications. If the length of filter employed in distributed network is very large, then the estimation of desired parameter takes more to evaluate the observations. Block-LMS is very useful when computational cost becomes more. The processing of adaption in Block-LMS is done block-by-block rather than sample-by-sample, which is normally used in all other types of LMS scheme. Here Block-Adaptive Filter is employed which contains both transform domain properties as well as block-by-block processing of data. Block Adaptive Filters are prepared by using transform like DFT, DCT, DHT and Overlap-Add DFT.Theoretically Block-LMS is most suitable as compare to other LMS schemes. Because it employs both transform domain characteristics which is beneficial in convergence performance of filter and other is block-by-block processing of data, which is beneficial in the sense of cost reduction. The computational cost reduction is determined by a factor which is greater than one. The cost reduction factor is a function of filter length and block size of data. Both these characteristics make the adaptive filter more suitable for real time environment.

The employment of Block Adaptive filter at each node must be done carefully because at each node Block Adaptive Filter creates many sub-band filters that is depends on the block size. When we work with only one Block Adaptive filter then it is very simple. But in case of distributed network where each node shares information with some neighboring nodes, useful care must be taken. Because at each node there are many sub-band filters and the information exchange becomes a challenge. So a particular sub-band filter must be share the information to the same positioned sub-band filter of neighboring node.

The Eigen value spread of a matrix is defined as the ratio of largest eigen value to smallest eigen value. Before applying transformation to the input, the eigen value spread is approximate equal to$(1+\rho)^2/(1-\rho)^2$. The eigen value spread after applying DFT transformation to input data and power normalization of input data is$(1+\rho)/(1-\rho)$. Where in case of DCT eigen value spread is $(1+\rho)$ [5].

CHAPTER 2

DISTRIBUTED NETWORKS & DISTRIBUTED STRATEGIES

A distributed network is a network consisting of many nodes that are located at different places. The nodes observe the parameter of interest from the environment and evaluate the desire parameter of interest locally by exchanging information with their neighboring nodes. The connectivity between the nodes are characterized by basically methods.

1. Incremental strategy
2. Diffusion strategy
3. Probabilistic strategy

I have worked on first two schemes incremental and diffusion.

2.1 INCREMENTAL CO-OPERATIVE STRATEGY

A distributed network containing many nodes and nodes are exchanging information from other nodes by incremental co-operative scheme is analyzed. The nodes have to respond in real time situation in order to estimate desire parameter of interest. Each node is capable to estimate the desire parameter of interest at local level with the help of observations taken by itself and information provided by neighboring node. In incremental fashion one node is allowed to take information from last node and after local estimation same node has to send the information to the next node. The connection of nodes forms a close like structure in this co-operative scheme. These type of distributed networks are useful in the applications like linking PCs to each other, laptops linking, cell phones linking, sensors linking, and in control networks. Applications will range from sensor networks to precision agriculture, environment monitoring, disaster relief management, smart spaces, target localization, as well as medical applications [8]–[15].

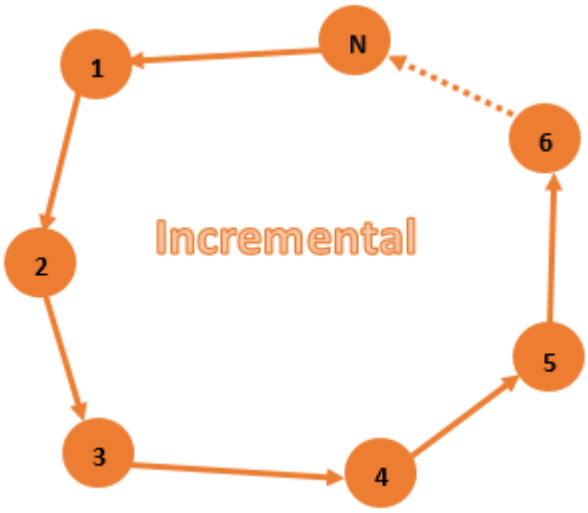


Figure-2. 1 Incremental strategy

As shown above each node is receiving information from previous node & sending to next node after local estimation. Now suppose a distributed network consisting of N nodes, connected via incremental co-operative scheme as shown above. The observations taken by nodes from the environment are aroused from different resources with different statistical profiles.

The mathematical equations to describe the entire process is as follows-

$$for\ iteration \quad i \geq 0, repeat$$

$$k = 1,2,3, \dots \dots N \qquad (k\ denotes\ k^{th} no.\ of\ node)$$

$$\psi_0^i = W_{i-1}$$

$$\psi_k^i = \psi_{k-1}^i + \mu_k u_{k,i}^* \left(d_k(i) - u_{k,i}\psi_{k-1}^i\right), \quad k = 1,2,3, \dots \dots N \dots \dots \dots \dots \quad (2.1.1)$$

$$W_i = \psi_n^i$$

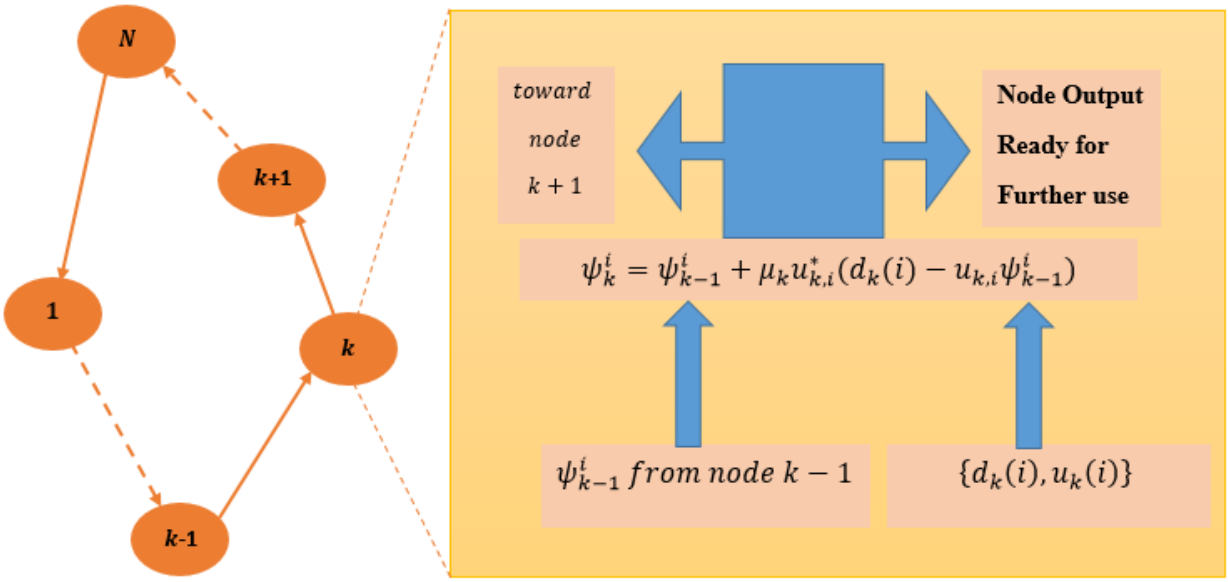


Figure-2. 2 Processing by node k in Incremental co-operation strategy

2.2 Diffusion Co-operative Strategy

A distributed network containing many nodes and nodes are exchanging information from other nodes by diffusion co-operative scheme is analyzed. The nodes have to respond in real time situation in order to estimate desire parameter of interest. Each node is capable to estimate the desire parameter of interest at local level with the help of observations taken by itself and information provided by pre-defined neighboring nodes. The share of information taken by a particular node from its predefined neighboring node is depends on total number of nodes connected to that particular node. Diffusion a very good scheme for distributed networks to work in real time phenomenon.

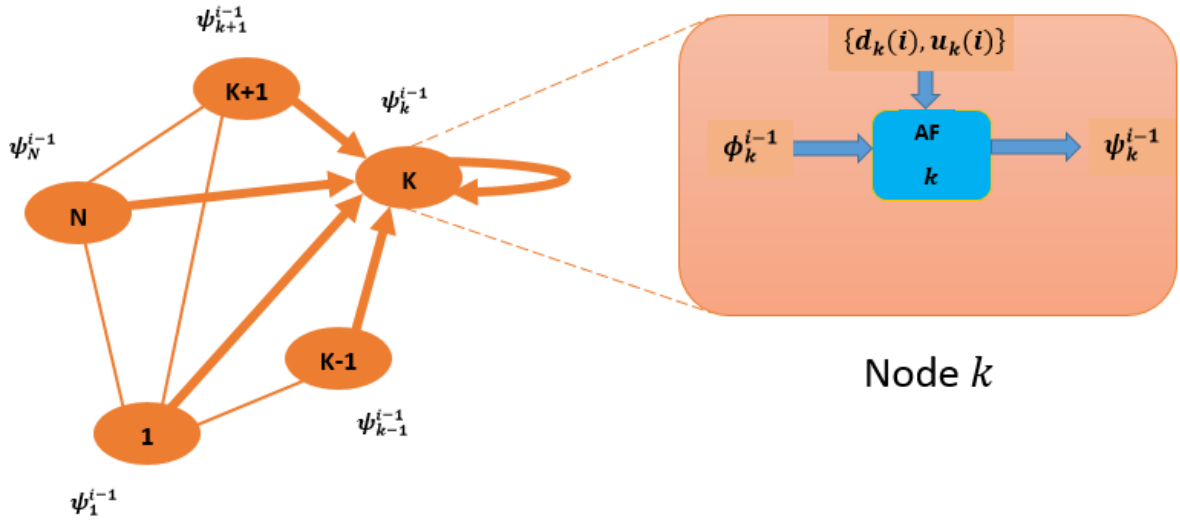


Figure-2. 3Processing by node k in diffusion co-operative scheme

I have worked on basically two topologies of diffusion which is shown below.

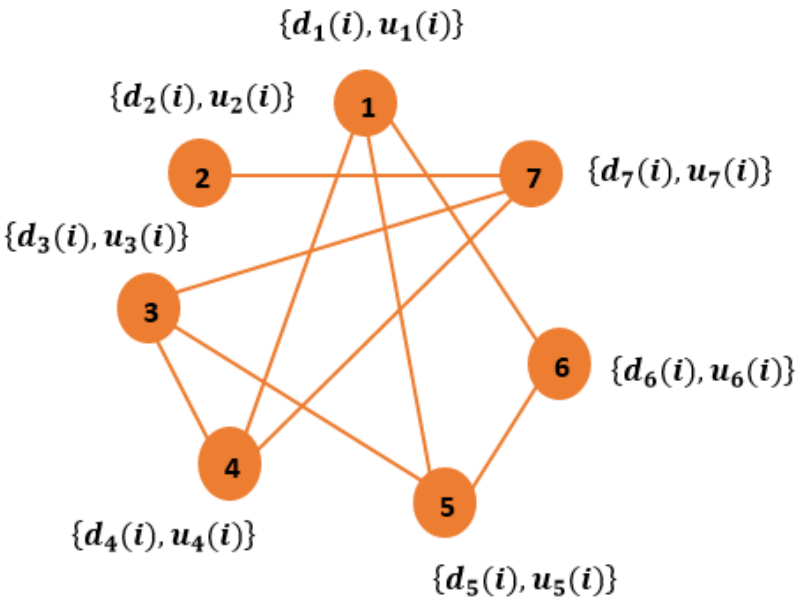


Figure-2. 4 A distributed network with 7 nodes in diffusion co-operation scheme

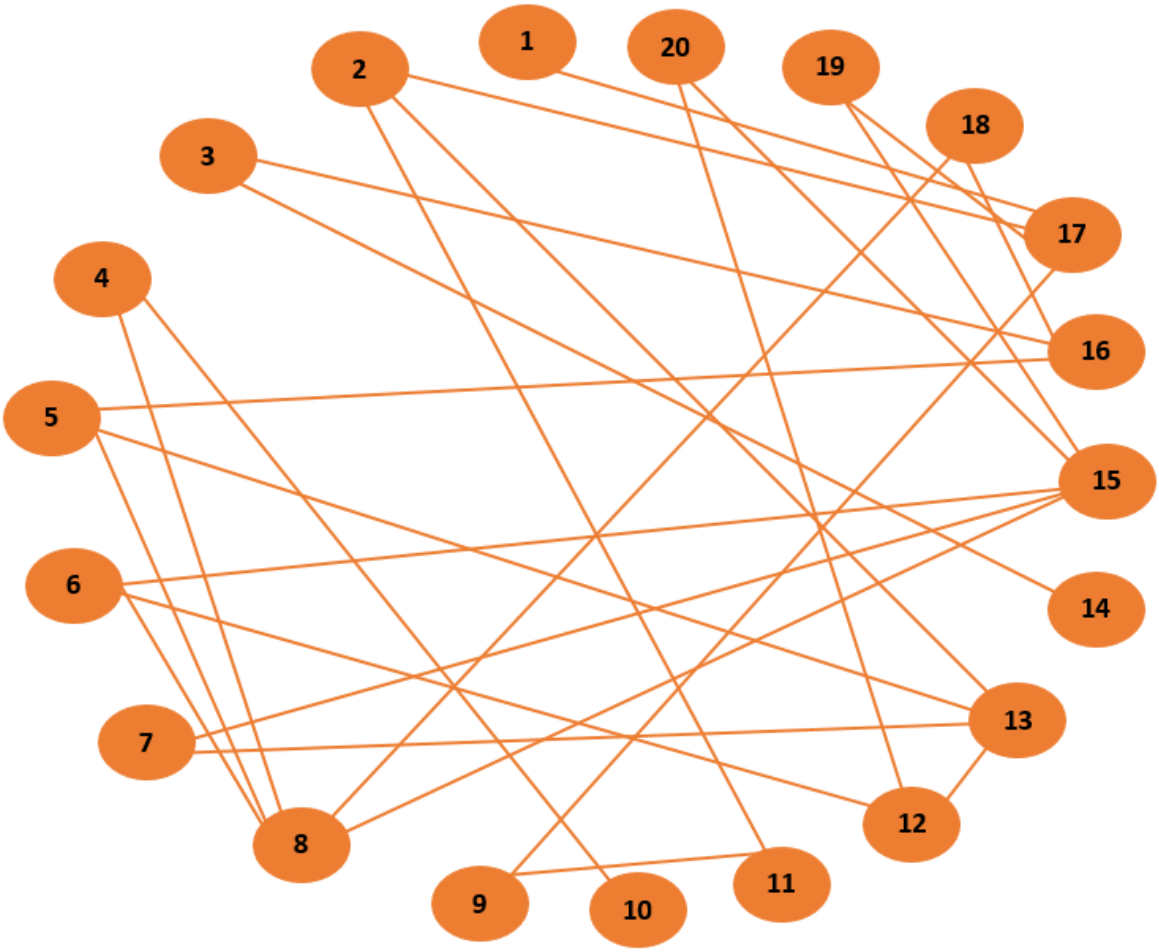


Figure-2. 5 A distributed network with 20 nodes in diffusion co-operation scheme

$$for\ iteration \quad i \geq 0, repeat$$

$$k = 1,2,3, \dots \dots N \qquad (k\ denotes\ k^{th} no.\ of\ node)$$

$$\phi_k^{(i-1)} = \sum_{l \epsilon N_{k,i-1}} C_{kl} \psi_l^{i-1}, \qquad \phi_k^{(-1)} = 0$$

$$\psi_k^{(i)} = \phi_k^{(i-1)} + \mu_k u_{k,i}^* \left(d_k(i) - u_{k,i} \phi_k^{(i-1)} \right) \dots \dots \dots \dots \quad (2.2.1)$$

CHAPTER 3

TRANSFORM DOMAIN & BLOCK ADAPTIVE FILTERING

3.1 Frequency Domain Processing by Unitary Transforms

We know that unitary transforms like DFT, DCT are well known for their orthogonal nature which is helpful in data de-correlation at the input stage. The characteristics of input is rarely known in co-relation sense [13] – [17]. The transform used in this scheme must full-fill $TT^* = T^*T = I$. Where T denotes the transform matrix of DFT and DCT. Two basic steps are followed in this mechanism-

1> The input regressor must be processed by transform matrix.

2> Power normalization of transformed input regressor.

The weight updating equation of Adaptive Filter is-

$$W_i = W_{i-1} + \mu u_i'(d(i) - u_i W_{i-1})$$

The length of filter is considered as M. So the size unitary transform matrix is $M \times M$.

The DFT matrix is defined as-

$$[F]_{km} = \frac{1}{\sqrt{M}} e^{-\frac{j2\pi mk}{M}}, \qquad k, m = 0,1,2, \dots \dots \dots M-1$$

Similarly the DCT matrix is defined as-

$$[C]_{km} = \alpha(k) \cos\left(\frac{k(2m+1)\pi}{2M}\right), \qquad k, m = 0,1,2, \dots \dots \dots M-1$$

Where

$$\alpha(0) = \frac{1}{\sqrt{M}} \qquad and \qquad \alpha(k) = \frac{2}{\sqrt{M}} \quad for\ k \neq 0$$

The transformed regressor is –

$$\bar{u}_i = u_i T$$

Which give the transformed regressor in DFT & DCT as-

Figure-3. 1 Transformed input regressor

The covariance matrix of transformed regressor is-

$$R_{\bar{u}} = E(\bar{u}_i * \bar{u}_i) = T^*(E(u_i * u_i))$$

$$R_{\bar{u}} = T^* R_u T$$

Now the weight matrix of filter is also processed by unitary transform matrix as-

$$\bar{W}_i = T^* W_i$$

Now the weight updating equation can be written as-

$$\bar{W}_i = \bar{W}_{i-1} + \mu \bar{u}_i^* (d(i) - \bar{u}_i W_{i-1}), \qquad \bar{W}_{-1} = 0$$

Now in order to proceed for power normalization process, means the input regressor is divide by input power to normalize.

Let's define a new term –

$$\lambda_k(i) = \beta \lambda_k(i-1) + (1-\beta)|\bar{u}_i(k)|^2 , \qquad k = 0,1,2, \dots \dots M-1$$

Where $0 \ll \beta < 1$.generally β is very close to one $\bar{u}_i(k)$ denotes the k-th entry of regressor $\bar{u}_i$

With the help of this power normalization factor, a diagonal matrix D is defined as –

$$D_i = diag\{\lambda_k(i)\}$$

Finally including all required concept, the weight updating equation becomes-

$$\bar{W}_i = \bar{W}_{i-1} + \mu D_i^{-1} \bar{u}_i^* e(i) \qquad \dots \dots \dots \dots (3.1.1)$$

3.1.1 Implementation of DFT-LMS

Suppose the length of filter is M. Similarly the input data also need to form in input regressor each of size M [13] – [17]. The construction of input regressor is like-

$$u_{i-1} = [u(i-1) \quad u(i-2) \quad \dots \dots \quad u(i-M+1) \quad u(i-M)]$$

And

$$u_i = [u(i) \quad u(i-1) \quad u(i-2) \quad \dots \dots \quad u(i-M+1)]$$

Now these input regressor are need to be processed by unitary transform matrix as-

$$\bar{u}_i(k) = \frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} u(i-m)\, e^{-\frac{j2\pi mk}{M}}$$

$$\bar{u}_{i-1}(k) = \frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} u(i-1-m)\, e^{-\frac{j2\pi mk}{M}}$$

Putting n=m+1

$$\bar{u}_{i-1}(k) = \frac{1}{\sqrt{M}} \sum_{n=1}^{M} u(i-n)\, e^{-\frac{j2\pi(n-1)k}{M}}$$

We got an interesting result –

$$\bar{u}_i(k) = e^{-\frac{j2\pi k}{M}} \bar{u}_{i-1}(k) + \frac{1}{\sqrt{M}}[u(i) - u(i-M)]$$

Let's define a new diagonal matrix S

$$S = \begin{bmatrix} 1 & & & & \\ & e^{-\frac{j2\pi}{M}} & & & \\ & & 1e^{-\frac{j4\pi}{M}} & & \\ & & & \ddots & \\ & & & & e^{-\frac{j2\pi(M-1)}{M}} \end{bmatrix}$$

Or

$$S = daig\left(1, \exp\left(-\frac{j2\pi}{M}\right), \ldots\ldots\ldots \exp\left(-\frac{j2\pi(M-1)}{M}\right)\right), \quad k = 0,1, \ldots . M-1$$

Now easily the input regressor is written in vector form as-

$$\bar{u}_i = \bar{u}_{i-1} S + \frac{1}{\sqrt{M}}[u(i) - u(i-M)][1 \quad 1 \ldots \ldots 1]$$

Let's $\lambda_k(-1) = very\ small + ve\ number,\ \bar{W}_{-1} = 0\ \ \&\ \bar{u}_{-1} = 0\ and\ repeat\ for\ i \geq 0$

$$\bar{u}_i = \bar{u}_{i-1} S + \frac{1}{\sqrt{M}}[u(i) - u(i-M)][1 \quad 1 \ldots \ldots 1]$$

$$\lambda_k(i) = \beta\lambda_k(i-1) + (1-\beta)|\bar{u}_i(k)|^2 \qquad k = 0,1, \ldots\ldots\ldots M-1$$

$$D_i = diag\{\lambda_k(i)\}$$

$$e(i) = d(i) - \bar{u}_i \bar{W}_{i-1}$$

$$\bar{W}_i = \bar{W}_{i-1} + \mu {D_i}^{-1} \bar{u}_i^*\ e(i) \ldots\ \ldots\ \ldots\ \ldots\ (3.1.2)$$

3.1.2 Implementation of DCT-LMS

Suppose the length of filter is M. Similarly the input data also need to form in input regressor each of size M [13] – [17]. The construction of input regressor is like-

$$u_{i-1} = [u(i-1) \quad u(i-2) \quad \dots\dots \quad u(i-M+1) \quad u(i-M)]$$

And $$u_i = [u(i) \quad u(i-1) \quad u(i-2) \quad \dots\dots \quad u(i-M+1)]$$

Now these input regressor are need to be processed by unitary DCT matrix and the relationship given as-

$$S = \begin{bmatrix} 2 & & & & \\ & 2\cos(\pi/M) & & & \\ & & 2\cos(2\pi/M) & & \\ & & & \ddots & \\ & & & & 2\cos((M-1)\pi/M) \end{bmatrix}$$

$$\bar{u}_i = \bar{u}_{i-1}S - \bar{u}_{i-2} + [\,\phi(0) \quad \phi(1) \dots\dots\dots \phi(M-1)]$$

Let's $\lambda_k(-1) = very\ small + ve\ number,\ \bar{W}_{-1} = 0$ & $\bar{u}_{-1} = 0$ $and\ repeat\ for\ i \geq 0$

$$a(k) = [u(i) - u(i-1)]\cos\left(\frac{k\pi}{2M}\right) \quad k = 0,1, \dots\dots M-1$$

$$b(k) = (-1)^k[u(i-M) - u(i-M-1)]\cos\left(\frac{k\pi}{2M}\right) \qquad k = 0,1, \dots\dots M-1$$

$$\phi(k) = \alpha(k)[a(k) - b(k)] \qquad k = 0,1, \dots\dots M-1$$

$$\bar{u}_i = \bar{u}_{i-1}S - \bar{u}_{i-2} + [\,\phi(0) \quad \phi(1) \dots\dots\dots \phi(M-1)]$$

$$\lambda_k(i) = \beta\lambda_k(i-1) + (1-\beta)|\bar{u}_i(k)|^2 \qquad k = 0,1, \dots\dots\dots M-1$$

$$D_i = diag\{\lambda_k(i)\}$$

$$e(i) = d(i) - \bar{u}_i\bar{W}_{i-1}$$

$$\bar{W}_i = \bar{W}_{i-1} + \mu D_i^{-1}\bar{u}_i^{*}e(i) \dots\dots\dots\dots (3.1.3)$$

3.2 Block Adaptive Filters

In transform domain adaptive filtering, the convergence issue of LMS is rectified because of input data de-correlation by unitary transforms like DCT & DFT. When the issue is related to computational cost, Block Adaptive Filters are preferred. Because Block Adaptive Filter utilize

Block LMS & also the unitary transforms like DCT & DFT. In Block LMS the data processing is done block by block rather than sample by sample.

Consider a FIR channel of length M. suppose the channel is excited by a zero mean random sequence.

$\boldsymbol{u}_i = [u(i)\ \ u(i-1)\ \ u(i-2)\ \ \dots\dots\dots\dots u(i-M+1)]$

$\boldsymbol{d}(i) = \boldsymbol{u}_i g + v(i)$

$$G(z) = g(0) + g(1)z^{-1} + g(2)z^{-2} + \ \dots\dots\dots g(M-1)z^{-M+1} = \sum_{k=0}^{M-1} g(k)z^{-k}$$

$\hat{d}(i) = u_i w_{i-1}$

$e(i) = d(i) - \hat{d}(i)$

$w_i = w_{i-1} + \mu u_i^* e(i)$ ... ... (3.2.1)

The above set-up requires $O(M)$ number of operations per sample. But when M is too large in size then the cost of this implementation is very large & prohibitive. In these kind of situations adaptive implementation is preferred [18] – [20].

In this adaptive implementation by block adaptive method, the process is very similar to LMS. The error $\{e(i)\}$& the estimate $\{d(i)\}$ is calculated in appropriate manner. The input data is converted to transformed regressor & processing is done in block-by-block manner rather than sample-by-sample manner. The resultant computation cost is reduced by a factor which is greater than one. Block adaptive filter has also better convergence than simple LMS because the eigenvalue spread of covariance matrix of transformed input regressor is reduced as compared to original input data.

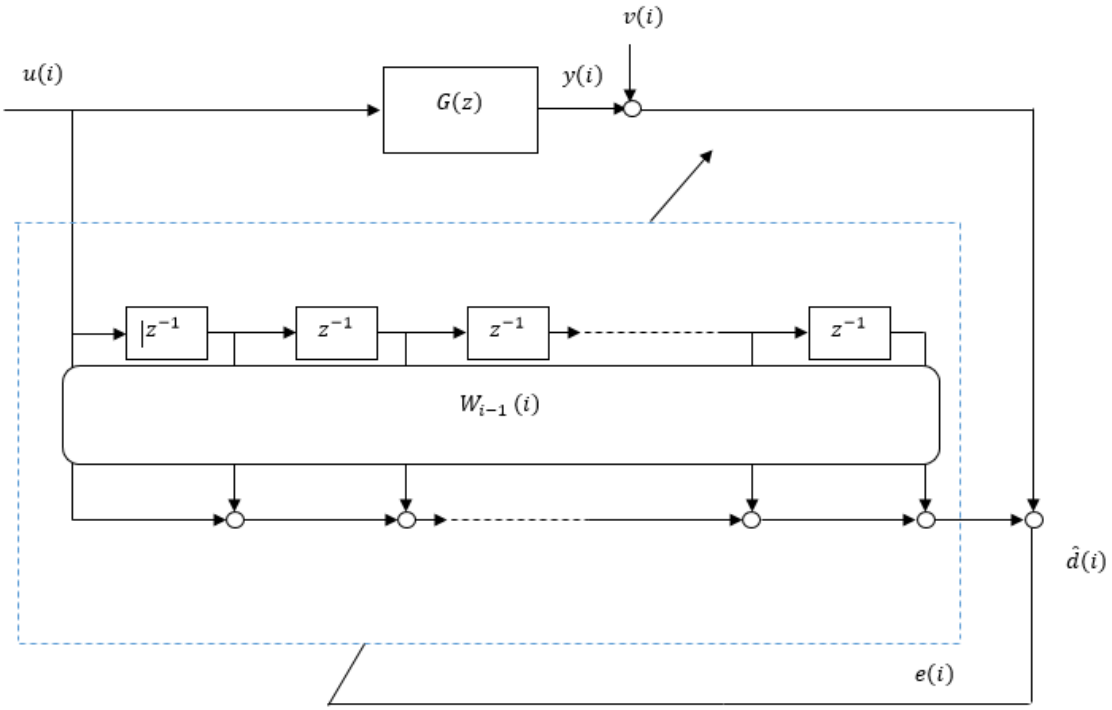


Figure-3. 2 Mechanism of Transform Domain Filtering

3.2.1 Block Convolution

Suppose there is a long FIR filter. The long length of filter is responsible for high computational cost. We need to convert this long length into some blocks of small length without doing adaptation process [13] – [17]. So first of all block implementation process should be done.

Now consider a long FIR filter having impulse response g & its transfer function $G(z)$, z-transform of input and output as $\{U(z), Y(z)\}$. Input and output in time domain are$\{u(i), y(i)\}$.

$$Y(z) \triangleq \sum_{i=0}^{\infty} y(i)\, z^{-i}$$

$$U(z) \triangleq \sum_{i=0}^{\infty} u(i)\, z^{-i}$$

The relationship of input and output with filter in z-domain is-

$$Y(z) = G(z)U(z)$$

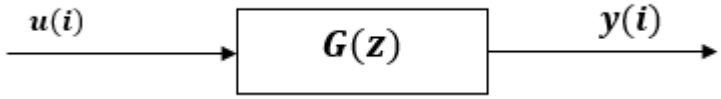


Figure-3. 3 general input/output relationship

A block is consist of several samples. Block processing means, processing of several samples at the same time. Hence it is called block processing.

The input samples must be converted into blocks (suppose length of block is B), and we get same length blocks at the output stage. The input and output in block vector as-

$$u_{B,n} \triangleq \begin{bmatrix} u(nB+B-1) \\ \vdots \\ u(nB+1) \\ u(nB) \end{bmatrix} \qquad y_{B,n} \triangleq \begin{bmatrix} y(nB+B-1) \\ \vdots \\ y(nB+1) \\ y(nB) \end{bmatrix}$$

Where B is block size & n is block index (no of blocks)

For example – the block size B is 3, the relation of block vectors of input/output $\{u_{B,n}, y_{B,n}\}$ with input/output sequence$\{u(i), y(i)\}$ as-

$$u(0)\ u(1)\ u(2) \rightarrow u_{3,0} \qquad y(0)\ y(1)\ y(2) \rightarrow y_{3,0}$$

$$u(3)\ u(4)\ u(5) \rightarrow u_{3,1} \qquad y(3)\ y(4)\ y(5) \rightarrow y_{3,1}$$

$$u(6)\ u(7)\ u(8) \rightarrow u_{3,2} \qquad y(6)\ y(7)\ y(8) \rightarrow y_{3,2}$$

$$\vdots \qquad\qquad \vdots$$

The input and output block vectors in z-domain can be represented as$\{U_B(z)Y_B(z)\}$.

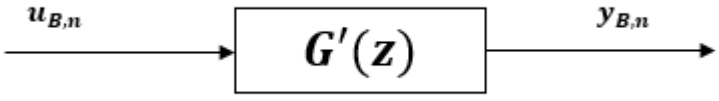


Figure-3. 4 Input/output relationship in block manner

The above relationship can be represented in mathematical form as-

$$U_B(z) \triangleq \sum_{n=0}^{\infty} u_{B,n} z^{-n} \qquad Y_B(z) \triangleq \sum_{n=0}^{\infty} y_{B,n} z^{-n}$$

The block processing with input and output is done above in both domain. Now we have to deal with long length FIR filter$G(z)$. The block size is B, hence we need to find B poly-phase component of FIR filter. Suppose the length of FIR filter is M. So each poly-phase component is of length M/B.

$$P_0(z) = g(0) + g(B)z^{-1} + g(2B)z^{-2} + \cdots$$

$$P_1(z) = g(1) + g(B+1)z^{-1} + g(2B+1)z^{-2} + \cdots$$

$$P_3(z) = g(2) + g(B+2)z^{-1} + g(2B+2)z^{-2} + \cdots$$

$$\vdots$$

$$P_{B-1}(z) = g(B-1) + g(2B-1)z^{-1} + g(3B-1)z^{-2} + \cdots$$

Here it is observed that first B coefficients of FIR filter transfer function $G(z)$ becomes the first coefficients of $P_k(z)$ [13] – [17]. The second B coefficients of $G(z)$ becomes the second coefficients of $P_k(z)$ and so on.

For example- B=3 and M=12, there will be three poly-phase components as-

$$P_0(z) = g(0) + g(3)z^{-1} + g(6)z^{-2} + g(9)z^{-3}$$

$$P_1(z) = g(1) + g(4)z^{-1} + g(7)z^{-2} + g(10)z^{-3}$$

$$P_2(z) = g(2) + g(5)z^{-1} + g(8)z^{-2} + g(11)z^{-3}$$

We can also represent $G'(z)$ in terms of $P_k(z)$. For e.g. B=3

$$G'(z) = \begin{bmatrix} P_0(z) & P_1(z) & P_2(z) \\ z^{-1}P_2(z) & P_0(z) & P_1(z) \\ z^{-1}P_1(z) & z^{-1}P_2(z) & P_0(z) \end{bmatrix} \ldots\ \ldots\ \ldots\ \ldots\ (3.2.2)$$

Here $G'(z)$ is a pseudo circulant matrix. Generally when all element below the diagonal elements of a circulant matrix is multiplied by z^{-1} is known as pseudo circulant matrix. The input/output block vector can be related to filter in z-domain as-

$$Y_B(z) = G'(z)U_B(z) \ldots\ \ldots\ \ldots\ \ldots\ (3.2.3)$$

The implementation of $G(z)$ is still not efficient. We need to define $G'(z)$ further specifically for practical point of view. $G'(z)$ Can be further factored as-

$$G'(z) = P'(z)Q(z) \dots \dots \dots \dots (3.2.4)$$

Here $P'(z)$ is $B \times 2B - 1$ matrix function which is a Toeplitz structure. Generally a circulant matrix is called as Toeplitz when identical entries along the diagonals with first row is circularly shifted to the right, one shift at a time in order to form other rows. For e.g. B=3

$$P'(z) = \begin{bmatrix} P_0(z) & P_1(z) & P_2(z) & 0 & 0 \\ 0 & P_0(z) & P_1(z) & P_2(z) & 0 \\ 0 & 0 & P_0(z) & P_1(z) & P_2(z) \end{bmatrix} \dots \dots \dots \dots (3.2.5)$$

Where $Q(z)$ is a $(2B-1) \times B$ matrix with two blocks (one upper block and other lower block) upper block is identity block and lower block is unit delay block. For e.g. B=3

$$Q(z) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \hline z^{-1} & 0 & 0 \\ 0 & z^{-1} & 0 \end{bmatrix} \dots \dots \dots \dots (3.2.6)$$

3.2.2 Block Convolution by Using DFT

Now the challenge is to use DFT in the block convolution scheme. When the work is done with DFT, the sequences are required to be power of two. So it is more suitable to define further $\{P'(z), Q(z)\}$ as-

$$P''(z) = \begin{bmatrix} P_0(z) & P_1(z) & P_2(z) & 0 & 0 & 0 \\ 0 & P_0(z) & P_1(z) & P_2(z) & 0 & 0 \\ 0 & 0 & P_0(z) & P_1(z) & P_2(z) & 0 \end{bmatrix} \dots \dots \dots \dots (3.2.7)$$

$$Q''(z) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \hline z^{-1} & 0 & 0 \\ 0 & z^{-1} & 0 \\ 0 & 0 & z^{-1} \end{bmatrix} \dots \dots \dots \dots (3.2.8)$$

The new dimensions of $\{P''(z), Q''(z)\}$ becomes $B \times 2B$ and $2B \times B$ respectively by padding a column of zeros to $P'(z)$ and row of zeros to $Q(z)$.

Formulation of Transfer Function-By using the matrices $\{P''(z), Q''(z)\}$ as defined above an another matrix $C(z)$ we define as-

$$C(z) = \left[\begin{array}{cccccc} P_0(z) & P_1(z) & P_2(z) & 0 & 0 & 0 \\ 0 & P_0(z) & P_1(z) & P_2(z) & 0 & 0 \\ 0 & 0 & P_0(z) & P_1(z) & P_2(z) & 0 \\ \hline 0 & 0 & 0 & P_0(z) & P_1(z) & P_2(z) \\ P_2(z) & 0 & 0 & 0 & P_0(z) & P_1(z) \\ P_1(z) & P_2(z) & 0 & 0 & 0 & P_0(z) \end{array}\right] \ldots \ldots \ldots \ldots (3.2.9)$$

Now $P''(z)$ can be gain from the top three rows of $C(z)$ as-

$$P''(z) = [I_B \quad 0_{B\times B}]C(z) \ldots \ldots \ldots \ldots (3.2.10)$$

Where I_B denotes the identity matrix of dimension $B \times B$ and $0_{B\times B}$ denotes the null matrix of dimension $\times B$.

Let's define the DFT matrix of dimension $2B \times 2B$ as-

$$[F]_{km} \triangleq e^{\frac{-j2\pi mk}{2B}} \qquad k, m = 0,1, \ldots 2B-1$$

A well-known result is that when any circulant matrix can be diagonalized by the DFT matrix [13] – [17]. In similar fashion the circulant matrix $C(z)$ can be diagonalized as-

$$C(z) = F^* L(z) F \ldots \ldots \ldots \ldots (3.2.11)$$

Where $C(z)$ is a diagonal matrix function which is defined as-

$$L(z) = \begin{bmatrix} L_0(z) & & & \\ & L_1(z) & & \\ & & \ddots & \\ & & & L_{2B-1}(z) \end{bmatrix} \ldots \ldots \ldots \ldots (3.2.12)$$

Where $L_k(z)$ represents the series of sub-band FIR filters with length of each filter is M/B. We can say that the first row of $C(z)$ can be related to the diagonal entries of $L(z)$. For e.g. B=3, the relation can be defined that-

$$\begin{bmatrix} P_0(z) \\ P_1(z) \\ P_2(z) \\ 0 \\ 0 \\ 0 \end{bmatrix} = F \begin{bmatrix} L_0(z) \\ L_1(z) \\ L_2(z) \\ L_3(z) \\ L_4(z) \\ L_5(z) \end{bmatrix} \ldots\ \ldots\ \ldots\ \ldots \quad (3.2.13)$$

The above relation tells that poly-phased component can mapped into the diagonal component $L(z)$ and vice-versa.

Combining the equations and we get-

$$G''(z) = [I_B \quad 0_{B\times B}]C(z)Q''(z)$$

$$G''(z) = [I_B \quad 0_{B\times B}]F^*L(z)FQ''(z)$$

The above mathematical representation can represented by diagram as-

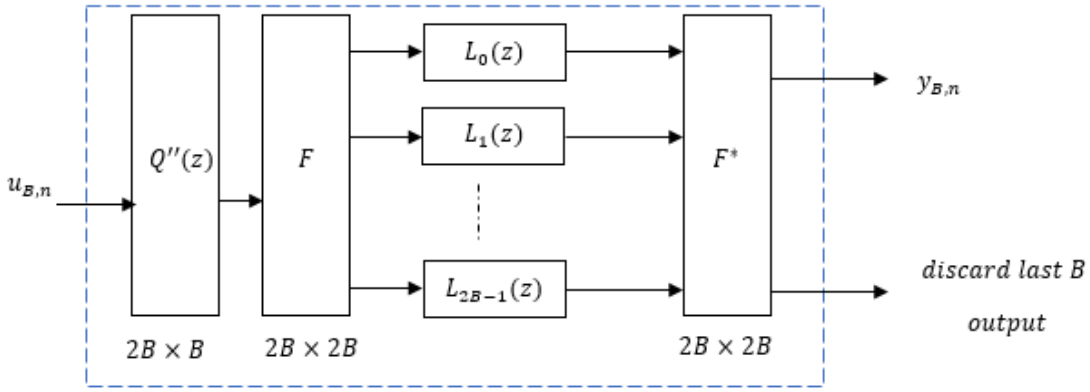


Figure-3. 5 Block convolution

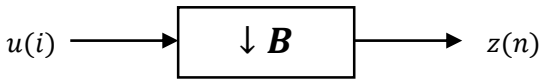
In time domain formulation the first matrix dimension is $2B \times B$. Hence we need to rearrange the input block vector to make $2B$ columns of input vector as-

$$u_{2B,n} = \begin{bmatrix} u_{B,n} \\ u_{B,n-1} \end{bmatrix}$$

For example: - at n=1

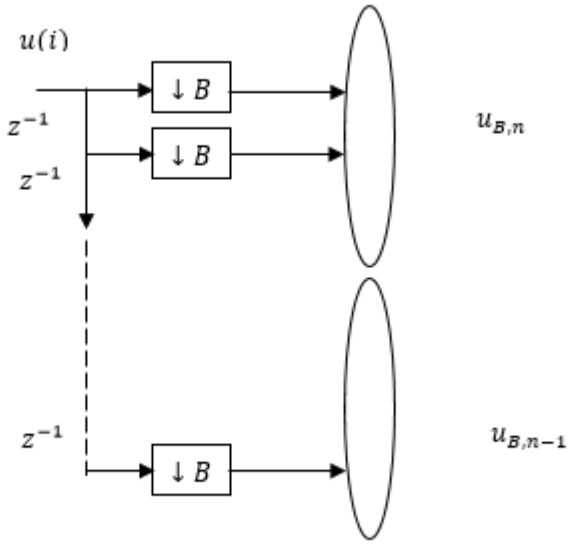
$$u_{2B,1} = \begin{bmatrix} u(5) \\ u(4) \\ u(3) \\ \hline u(2) \\ u(1) \\ u(0) \end{bmatrix}$$

The real effect of $Q''(z)$ is to convert serial to parallel data. In other way we can also represents the serial to parallel conversion as follows. Let $\downarrow B$ denotes the decimator of order B, we can say that-



Where the relation between input and output is- $z(n) = u(nB)$, $n = 0,1,2 \ldots \ldots$

Here n represents lower rate signal and i represents higher rate signal. The input block vector of size B can constructed by the above decimation implementation. After the block formation we have to put two consecutive blocks in a column which results a block vector $\{u_{B,n}, u_{B,n-1}\}$



.

Figure-3. 6 Formation of two consecutive input blocks

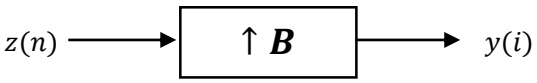
Figure: using $2B$ decimator formulation of block data vector $col\{u_{B,n}, u_{B,n-1}\}$

After dealing with matrix $Q''(z)$ the resulting block vector processed by DFT matrix and hence resulting vector becomes $2B \times 1$ transformed vector as-

$$u'_{2B,n} \triangleq \begin{bmatrix} u'_0(n) \\ u'_1(n) \\ \vdots \\ \vdots \\ u'_{2B-1}(n) \end{bmatrix} = F \begin{bmatrix} u_{B,n} \\ u_{B,n-1} \end{bmatrix}$$

Now the entries $u'_{2B,n}$ are fed into the series of sub-band filters $\{L_k(z)\}$ and the resulting outputs are processed by the conjugated DFT matrixF^*. In the final response top B outputs are kept and lower B outputs are discarded.

As the decimation process is done at the input stage to form samples into blocks, in the same fashion the response at output stage can be interpolated with the help of $\uparrow B$ interpolater. The interpolator convert the output data from parallel to serial fashion. The representation of functioning of interpolator is shown below.



The mathematical representation of the above diagram is-

$$y(i) = \begin{cases} z\left(\frac{i}{B}\right) & if \frac{i}{B} \; is \; an \; interger \\ 0 & otherwise \end{cases}$$

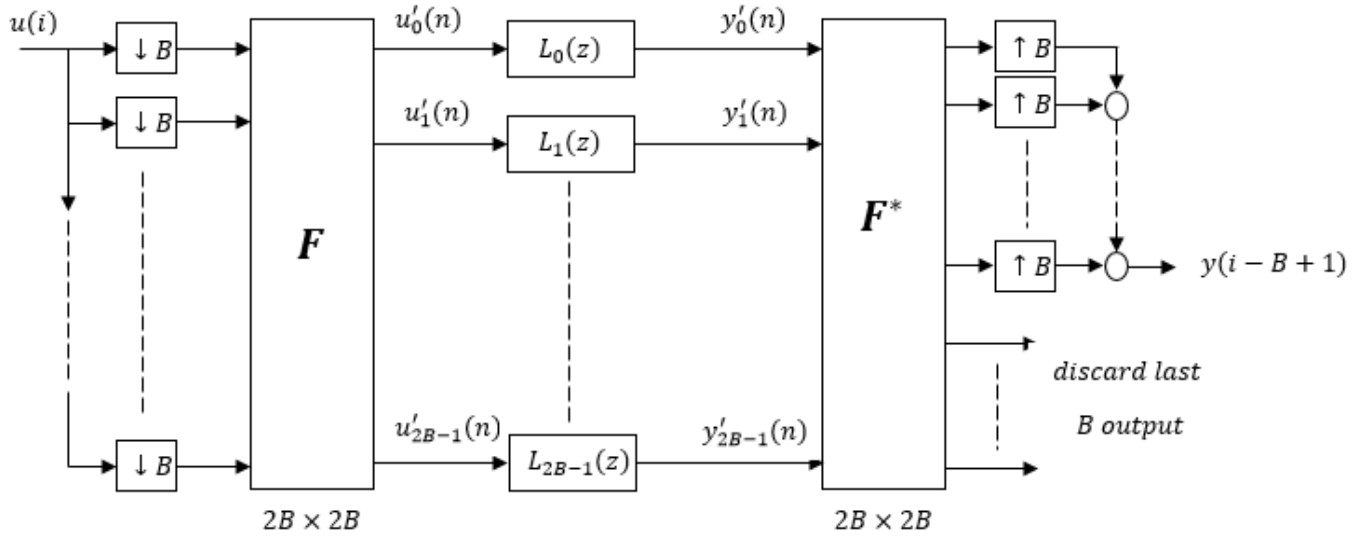


Figure-3. 7 Block convolution full

Computational complexity-of block convolution has three steps as-

Step -1:- The first transformed vector contains-

$$F\begin{bmatrix} u_{B,n} \\ u_{B,n-1} \end{bmatrix}$$

It requires total number of $B \log_2 2B$ complex multiplications.

Step-2:- Total number of 2B filters are present and each filter is having length M/B. Each filter requires M/B number of inner products. So total no of complex multiplications are $2B * \frac{M}{B} = 2M$.

Step-3:- At last transform stag 2B outputs are generated. It requires total number of $B \log_2 2B$ complex multiplications.

For step 1-3, a total number of $2M + 2B \log_2 2B$ complex multiplications is required for each block of input of size B.

The filter $L_k(z)$ is also calculated from $G(z)$ as-

$$\begin{bmatrix} L_0(z) \\ L_1(z) \\ \vdots \\ L_{2B-1}(z) \end{bmatrix} = \frac{1}{2B} F^* \begin{bmatrix} P_0(z) \\ \vdots \\ P_{B-1}(z) \\ 0_{B\times 1} \end{bmatrix}$$

For e.g. B=3, M=12

$$\begin{bmatrix} l_{00} & l_{01} & l_{02} & l_{03} \\ l_{10} & l_{11} & l_{12} & l_{13} \\ l_{20} & l_{21} & l_{22} & l_{23} \\ l_{30} & l_{31} & l_{32} & l_{33} \\ l_{40} & l_{41} & l_{42} & l_{43} \\ l_{50} & l_{51} & l_{52} & l_{53} \end{bmatrix} = \frac{1}{6} F^* \begin{bmatrix} g(0) & g(3) & g(6) & g(9) \\ g(1) & g(4) & g(7) & g(10) \\ g(2) & g(5) & g(8) & g(11) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The computational cost required for above step is M/B DFT's of size 2B each. The total cost becomes $M \log_2(2B)$ complex multiplications. But in adaptive filter implementation phase $L_k(z)$ is updated in every iteration. If the size of block is normalized, there will be $\frac{M}{B}\log_2(2B)$ complex multiplication per input sample.

Finally the total cost associated with time domain block convolution is-

$$\frac{2M}{B} + \left(\frac{M}{B} + 2\right)\log_2(2B)$$

Finally the conclusion is that –

1> The direct convolution method requires $O(M)$ operations per sample.

2> Frequency domain implementation requires $O(2M/B)$ operations per sample.

3> The reduction in computational complexity is decided by the block size B.

4> But the larger block size B also results to delay in signal path.

Comparison of computational complexity-

$$\frac{Frequency\ domain\ implementation\ cost}{direct\ convolution\ implementation\ cost} = \frac{\frac{2M}{B} + \left(\frac{M}{B}+2\right)\log_2(2B)}{M} \quad \ldots\ \ldots\ \ldots \quad (3.2.14)$$

3.2.3 DFT Unconstrained Block Adaptive filter:-

As discussed above the block adaptive implementation requires block-by-block processing with transform domain implementation of data. Both of these combined technique give better computational cost reduction as well as better convergence performance. Let's make the analysis step-by-step;-

Step 1> Convert the input repressor and noise signal in blocks and then make the two consecutive input block vector into block column-wise

$$u_{B,n} \triangleq \begin{bmatrix} u(nB+B-1) \\ \vdots \\ u(nB+1) \\ u(nB) \end{bmatrix} \qquad v_{B,n} = \begin{bmatrix} v(nB+B-1) \\ \vdots \\ v(nB+1) \\ v(nB) \end{bmatrix}$$

$$u_{2B,n} = \begin{bmatrix} u_{B,n} \\ u_{B,n-1} \end{bmatrix} \dots \dots \dots \dots \quad (3.2.15)$$

Step 2- Now make the input block vector into transformed regressor by the DFT matrix-

$$u'_{2B,n} = F * u_{2B,n} = col\{u'_k(n), \qquad k = 0,1, \dots \dots \dots, 2B-1\} \dots \dots \dots \dots \quad (3.2.16)$$

Where $u'_k(n)$ is a transformed regress foe each sub-band filter and length of $u'_k(n)$ for each k is M/B, which is represented as-

$$u'_k(n) = \left[u'_k(n) \dots \dots \dots \; u'_k\left(n - \frac{M}{B} + 1\right)\right], \qquad k = 0,1,2, \dots \dots \dots 2B-1$$

Step 3- The FIR filter of length B is converted into 2B sub-band filters, each having length M/B by the following relation (also explained above)-

$$\begin{bmatrix} L_0(z) \\ L_1(z) \\ \vdots \\ L_{2B-1}(z) \end{bmatrix} = \frac{1}{2B} F^* \begin{bmatrix} P_0(z) \\ \vdots \\ P_{B-1}(z) \\ 0_{B\times 1} \end{bmatrix} \dots \dots \dots \dots \quad (3.2.17)$$

Step 4- The desired output is calculated as structure of filter is depicted and converted into block vector of size B.

$$y'_k(n) = u'_{k,n} L_k, \qquad k = 0,1, \dots \dots \dots, 2B-1$$

$$d_{B,n} = [I_B \quad 0_{B\times B}] F^* y'_k(n) + v_{B,n}$$

Step 5- The actual output is also calculated in same fashion as-

$$y1'_k(n) = u'_{k,n} l_{k,n-1}, \qquad k = 0,1, \dots \dots \dots, 2B-1$$

$$\hat{d}_{B,n} = [I_B \quad 0_{B\times B}] F^* y1'_k(n)$$

Step 6- The error vector is difference of desired output vector and actual output vector-

$$e_{B,n} = d_{B,n} - \hat{d}_{B,n}$$

Step 7- This error vector is processed by DFT matrix and last B outputs are neglected as-

$$e'_{2B,n} = F \begin{bmatrix} I_B \\ 0_{B\times B} \end{bmatrix} e_{B,n} = col\{e'_k(n), \qquad k = 0,1, \dots \dots \dots 2B-1\}$$

Step 8- The weight of each sub-band filter is updated separately with normalized Block LMS as-

$$l_{k,n} = l_{k,n-1} + \frac{\mu}{\lambda_k} u'^{*}_{k,n} e'_k(n), \qquad k = 0,1, \dots \dots \dots 2B-1 \dots \dots \dots \dots \quad (3.2.18)$$

Where is calculated as- $\lambda_k(n) = \beta \lambda_k(n-1) + (1-\beta)|u'_k(n)|^2$

Step 9-The actual output block vector and error block vector are interpolated with the help of size B interpolator as-

$$\hat{d}_{B,n} = col\{\hat{d}(nB+B-1), \dots \dots \dots, \hat{d}(nB+1), \hat{d}(nB)\}$$

$$e_{B,n} = col\{e(nB+B-1), \dots \dots \dots, e(nB+1), e(nB)\}$$

Step 10- Mean square error is calculated by the output of interpolator of error vector-

$$mse = e(i) .* e(i)$$

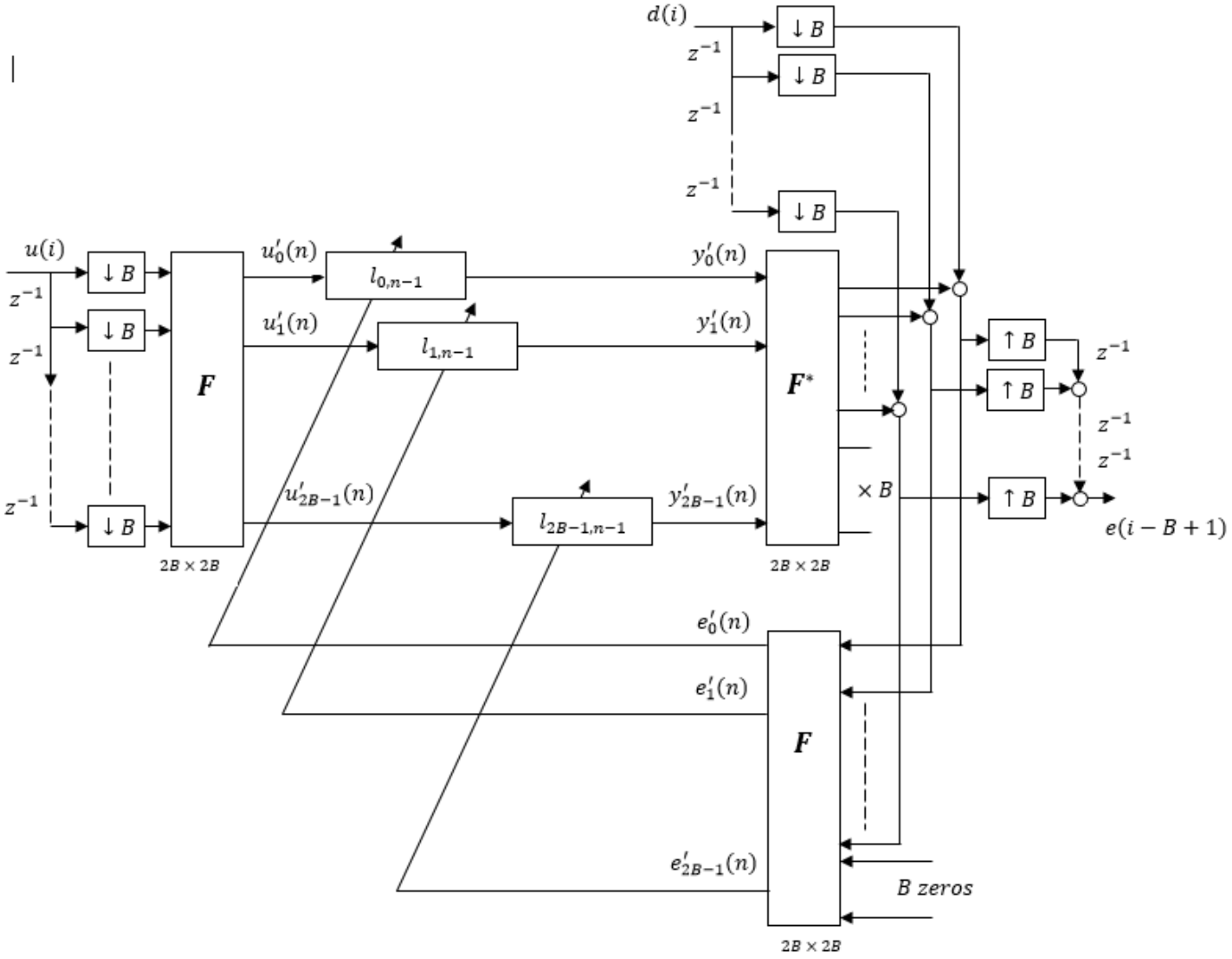


Figure-3. 8 Unconstrained DFT Block Adaptive Filter

3.2.4 DFT Constrained Filter Implementation

While computing for above DFT unconstrained scheme, we observe that the computation of $l_{k,n}$ is not satisfying the constraint of eq-**222.** It seems that for each iteration in time domain the following equation is not giving matrix whose last B rows are zero.

$$F\begin{bmatrix} l_{0,n}^T \\ l_{1,n}^T \\ \vdots \\ l_{2B-1,n}^T \end{bmatrix}$$

To make the arrangement constrained equation - (3.2.13) must be satisfied after every iteration n, and it can be possible if multiply equation-(3.2.13) by $\begin{bmatrix} I_B & \\ & 0_{B\times B} \end{bmatrix}$.

It means the arrangement seems to be like- $\begin{bmatrix} \times \\ 0_{B\times B} \end{bmatrix} = \begin{bmatrix} I_B & \\ & 0_{B\times B} \end{bmatrix} F \begin{bmatrix} l_{0,n}^T \\ l_{1,n}^T \\ \vdots \\ l_{2B-1,n}^T \end{bmatrix}$

We can also express the above relation in poly-phase component form as-

$$\begin{bmatrix} p_{0,n}^T \\ p_{1,n}^T \\ \vdots \\ p_{B-1,n}^T \\ \hline 0_{B\times B} \\ \\ \\ \end{bmatrix} = \begin{bmatrix} I_B & \\ & 0_{B\times B} \end{bmatrix} F \begin{bmatrix} l_{0,n}^T \\ l_{1,n}^T \\ \vdots \\ l_{2B-1,n}^T \end{bmatrix} \dots \dots \dots \dots \quad (3.2.19)$$

In similar fashion we can also calculate the reverse by applying the following relation as-

$$\begin{bmatrix} l_{0,n}^{cT} \\ l_{1,n}^{cT} \\ \vdots \\ l_{2B-1,n}^{cT} \end{bmatrix} = \frac{1}{2B} F^* \begin{bmatrix} p_{0,n}^T \\ p_{1,n}^T \\ \vdots \\ p_{B-1,n}^T \\ \hline 0_{B\times B} \\ \\ \\ \end{bmatrix}$$

The new estimates $l_{k,n}^{cT}$ are satisfying the product

$$F\begin{bmatrix} l_{0,n}^{cT} \\ l_{1,n}^{cT} \\ \vdots \\ l_{2B-1,n}^{cT} \end{bmatrix}$$

Gives last B rows zeros in the resultant matrix.

The whole process of unconstrained DFT implementation is repeated in constrained DFT block adaptive filter as above, but finally the sub-band weights are modified as-

$$\begin{bmatrix} l_{0,n}^{cT} \\ l_{1,n}^{cT} \\ \vdots \\ l_{2B-1,n}^{cT} \end{bmatrix} = \frac{1}{2B} F^* \begin{bmatrix} I_B & \\ & 0_{B\times B} \end{bmatrix} F \begin{bmatrix} l_{0,n}^{T} \\ l_{1,n}^{T} \\ \vdots \\ l_{2B-1,n}^{T} \end{bmatrix} \ldots \ldots \ldots \ldots \quad (3.2.20)$$

Other all parameter calculation is same.

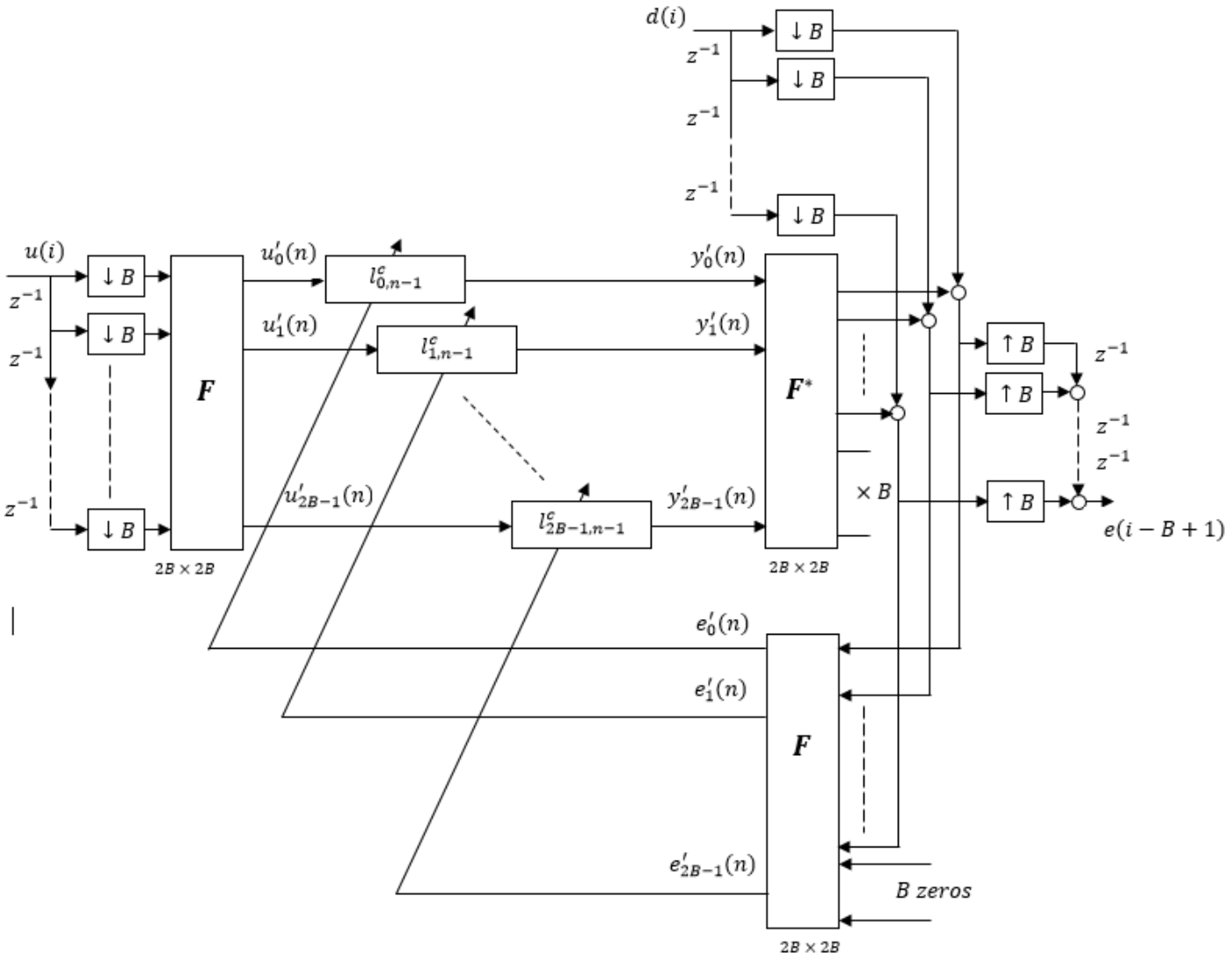


Figure-3. 9 Constrained DFT Block Adaptive Filter

3.2.5 Overlap-Add DFT Block Adaptive Filter

As DFT block implementation discussed above, in similar way all the terms can be calculated for overlap-Add Block Adaptive Filter.

$$G'(z) = \bar{Q}_1(z)\bar{P}_1(z)$$

For e.g. B=3

$$\bar{P}_1(z) = \begin{bmatrix} 0 & 0 & 0 \\ P_2(z) & 0 & 0 \\ P_1(z) & P_2(z) & 0 \\ P_0(z) & P_1(z) & P_2(z) \\ 0 & P_0(z) & P_1(z) \\ 0 & 0 & P_0(z) \end{bmatrix}$$

$$\bar{Q}_1(z) = \begin{bmatrix} z^{-1} & 0 & 0 & 1 & 0 & 0 \\ 0 & z^{-1} & 0 & 0 & 1 & 0 \\ 0 & 0 & z^{-1} & 0 & 0 & 1 \end{bmatrix}$$

$$\bar{P}_1(z) = C(z)\begin{bmatrix} 0_{B\times B} \\ I_B \end{bmatrix}$$

$$G'(z) = \bar{Q}_1(z)C(z)\begin{bmatrix} 0_{B\times B} \\ I_B \end{bmatrix} = \bar{Q}_1(z)F^*L(z)F\begin{bmatrix} 0_{B\times B} \\ I_B \end{bmatrix}$$

$$\bar{Q}_1(z) = [0_{B\times B} \quad I_B] + z^{-1}[I_B \quad 0_{B\times B}]$$

$$G'(z) = [0_{B\times B} \quad I_B]F^*L(z)F\begin{bmatrix} 0_{B\times B} \\ I_B \end{bmatrix} + z^{-1}[I_B \quad 0_{B\times B}]F^*L(z)F\begin{bmatrix} 0_{B\times B} \\ I_B \end{bmatrix} \ldots\ \ldots\ \ldots\ \ldots \quad (3.2.21)$$

The relation between the matrix $F\begin{bmatrix} 0_{B\times B} \\ I_B \end{bmatrix}$ and $F\begin{bmatrix} I_B \\ 0_{B\times B} \end{bmatrix}$ is given as-

$$F\begin{bmatrix} I_B \\ 0_{B\times B} \end{bmatrix} = JF\begin{bmatrix} 0_{B\times B} \\ I_B \end{bmatrix}$$

Where J is a matrix of size$2B \times 2B$ which is having $\pm$ alternatively.

$$J = diag\{1, -1,1, -1, \dots \dots ,1, -1\}$$

Substituting the above relation in equation-

$$G'(z) = [0_{B\times B} \quad I_B]F^* L(z)(I + z^{-1})F \begin{bmatrix} 0_{B\times B} \\ I_B \end{bmatrix}$$

The new transformed input regression can be expressed as-

$$s'_k(n) = u'_k(n) + (-1)^k u'_k(n-1), \qquad k = 0,1,2, \dots \dots \dots 2B-1 \dots \dots \dots \dots \quad (3.2.22)$$

Unconstrained Overlap-Add DFT Block Adaptive filter:-

Step 1- Convert the input repressor and noise signal in blocks and then make the two consecutive input block vector into block column-wise

$$u_{B,n} \triangleq \begin{bmatrix} u(nB+B-1) \\ \vdots \\ u(nB+1) \\ u(nB) \end{bmatrix} \qquad v_{B,n} = \begin{bmatrix} v(nB+B-1) \\ \vdots \\ v(nB+1) \\ v(nB) \end{bmatrix}$$

$$u_{2B,n} = \begin{bmatrix} 0_{B\times 1} \\ u_{B,n} \end{bmatrix}$$

Step 2- Now make the input block vector into transformed regressor by the DFT matrix-

$$u'_{2B,n} = F * u_{2B,n} = col\{u'_k(n), \qquad k = 0,1, \dots \dots \dots ,2B-1\}$$

The input regressor block vector is obtained as-

$$s'_{2B,n} = u'_{2B,n} + Ju'_{2B,n-1} = col\{s'_k(n), \qquad k = 0,1,2, \dots \dots \dots 2B-1\}$$

Where $s'_k(n)$ is a transformed regressor for each sub-band filter and length of $s'_k(n)$ for each k is M/B, which is represented as-

$$s'_k(n) = \left[s'_k(n) \dots \dots \dots \; s'_k\left(n - \frac{M}{B} + 1\right)\right], \qquad k = 0,1,2, \dots \dots \dots 2B-1$$

Step 3- The FIR filter of length B is converted into 2B sub-band filters, each having length M/B by the following relation (also explained above)-

$$\begin{bmatrix} L_0(z) \\ L_1(z) \\ \vdots \\ L_{2B-1}(z) \end{bmatrix} = \frac{1}{2B} F^* \begin{bmatrix} P_0(z) \\ \vdots \\ P_{B-1}(z) \\ 0_{B\times 1} \end{bmatrix}$$

Step 4- The desired output is calculated as structure of filter is depicted and converted into block vector of size B.

$$y'_k(n) = s'_{k,n} L_k, \qquad k = 0,1, \dots \dots \dots, 2B-1$$

$$d_{B,n} = [I_B \quad 0_{B\times B}] F^* y'_k(n) + v_{B,n}$$

Step 5- The actual output is also calculated in same fashion as-

$$y1'_k(n) = s'_{k,n} l_{k,n-1}, \qquad k = 0,1, \dots \dots \dots, 2B-1$$

$$\hat{d}_{B,n} = [I_B \quad 0_{B\times B}] F^* y1'_k(n)$$

Step 6- The error vector is difference of desired output vector and actual output vector-

$$e_{B,n} = d_{B,n} - \hat{d}_{B,n}$$

Step 7- This error vector is processed by DFT matrix and last B outputs are neglected as-

$$e'_{2B,n} = F \begin{bmatrix} I_B \\ 0_{B\times B} \end{bmatrix} e_{B,n} = col\{e'_k(n), \qquad k = 0,1, \dots \dots \dots 2B-1\}$$

Step 8- The weight of each sub-band filter is updated separately with normalized Block LMS as-

$$l_{k,n} = l_{k,n-1} + \frac{\mu}{\lambda_k} s'^{*}_{k,n} e'_k(n), \qquad k = 0,1, \dots \dots \dots 2B-1 \dots \dots \dots \dots \quad (3.2.23)$$

Where is calculated as- $\lambda_k(n) = \beta\lambda_k(n-1) + (1-\beta)|s'_k(n)|^2$

Step 9- The actual output block vector and error block vector are interpolated with the help of size B interpolator as-

$$\hat{d}_{B,n} = col\{\hat{d}(nB+B-1), \dots \dots \dots, \hat{d}(nB+1), \hat{d}(nB)\}$$

$$e_{B,n} = col\{e(nB+B-1), \dots \dots \dots, e(nB+1), e(nB)\}$$

Step 10- Mean square error is calculated by the output of interpolator of error vector-

$$mse = e(i) .* e(i)$$

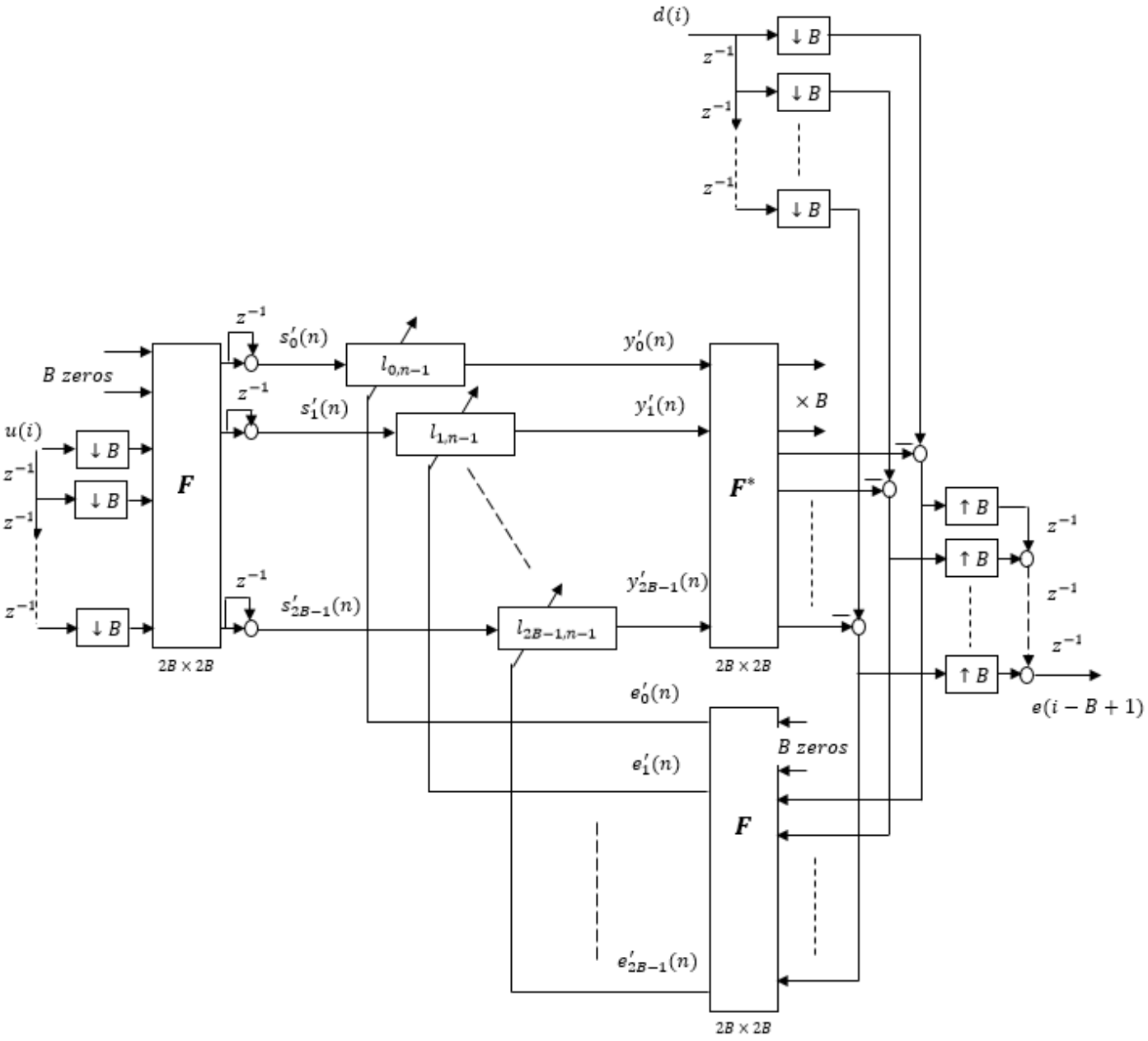


Figure-3. 10 Unconstrained Overlap-Add DFT Block Adaptive Filter

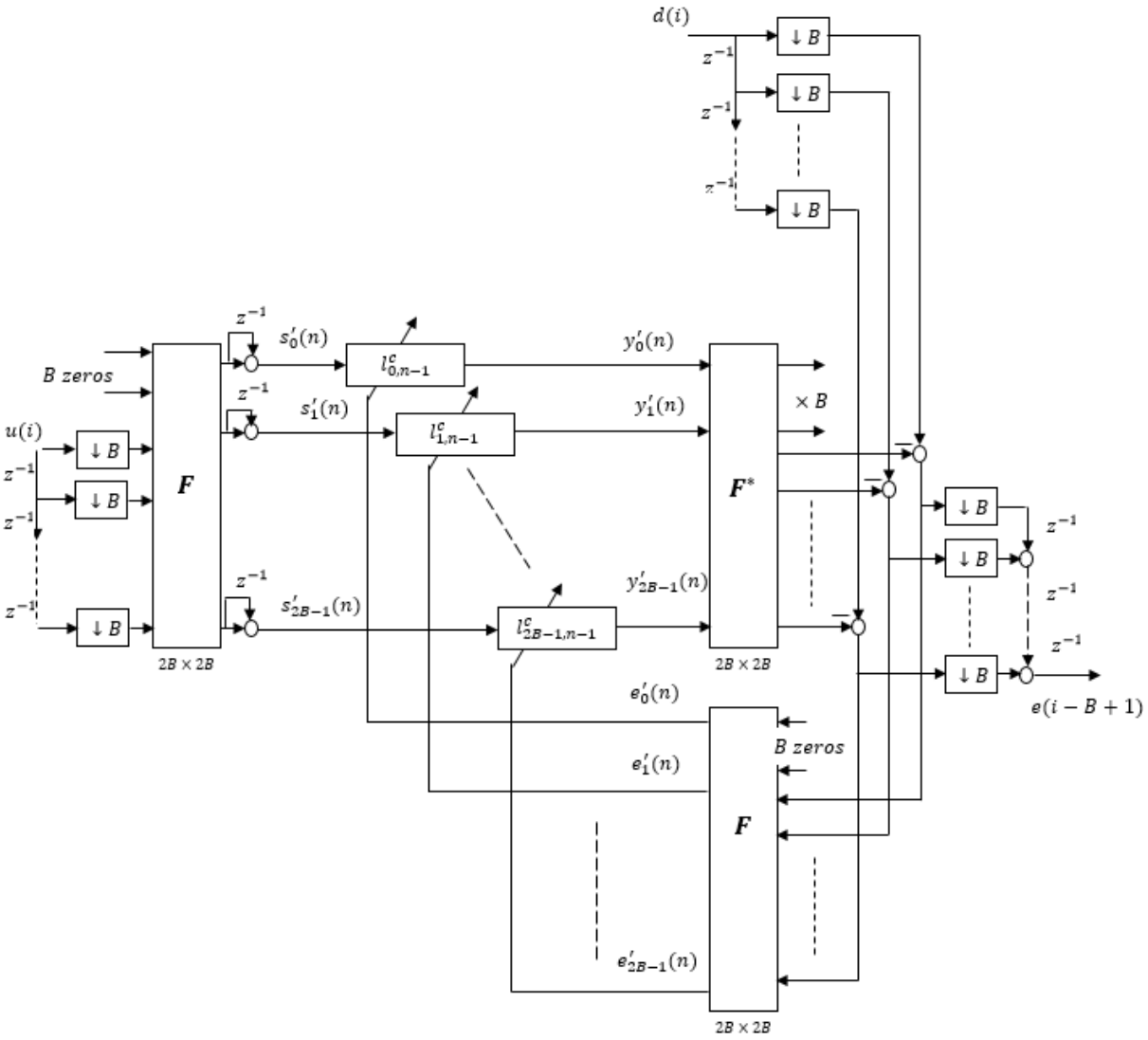


Figure-3. 11 Constrained Overlap-Add DFT Block Adaptive Filter

3.2.6 DHT Based Block Adaptive Filter

A DHT matrix is defined as-

$$[H]_{m,k} = \frac{1}{\sqrt{K}}\left[\cos\left(\frac{2mk\pi}{K}\right) - \sin\left(\frac{2mk\pi}{K}\right)\right], . \qquad m,k = 0,1,2,\dots\dots K-1$$

The Hartley matrix also satisfies both symmetric and orthogonal property as-

$$HH^T = H^2 = I$$

$$A(z) = \begin{bmatrix} a_0(z) & a_1(z) & a_2(z) & a_2(z) & a_1(z) \\ a_1(z) & a_0(z) & a_1(z) & a_2(z) & a_2(z) \\ a_2(z) & a_1(z) & a_0(z) & a_1(z) & a_2(z) \\ a_2(z) & a_2(z) & a_1(z) & a_0(z) & a_1(z) \\ a_1(z) & a_2(z) & a_2(z) & a_1(z) & a_0(z) \end{bmatrix} \dots\ \dots\ \dots\ \dots \quad (3.2.24)$$

$$P(z) = [I_B \quad 0_{B\times 2B}]A(z)\begin{bmatrix} 0_{1\times 2B-1} \\ I_{2B-1} \\ 0_{B\times 2B-1} \end{bmatrix} \dots\ \dots\ \dots\ \dots \quad (3.2.25)$$

.

$$A(z) = HL(z)H \dots\ \dots\ \dots\ \dots \quad (3.2.26)$$

$$L(z) = \begin{bmatrix} L_0(z) & & & \\ & L_1(z) & & \\ & & \ddots & \\ & & & L_{3B-1}(z) \end{bmatrix} \dots\ \dots\ \dots\ \dots \quad (3.2.26)$$

$$\begin{bmatrix} 0 \\ P_0(z) \\ \vdots \\ P_{B-1}(z) \\ 0_{B-1\times 1} \\ P_{B-1}(z) \\ \vdots \\ P_0(z) \end{bmatrix} = H\begin{bmatrix} L_0(z) \\ L_0(z) \\ \vdots \\ L_{3B-1}(z) \end{bmatrix} \dots\ \dots\ \dots\ \dots \quad (3.2.27)$$

$$G'(z) = [I_B \quad 0_{B\times 2B}]A(z)\begin{bmatrix}0_{1\times 2B-1}\\ I_{2B-1}\\ 0_{B\times 2B-1}\end{bmatrix}Q(z)$$

$$G'(z) = [I_B \quad 0_{B\times 2B}]HL(z)H\begin{bmatrix}0_{1\times 2B-1}\\ I_{2B-1}\\ 0_{B\times 2B-1}\end{bmatrix}Q(z)$$

$$G'(z) = [I_B \quad 0_{B\times 2B}]A(z)\begin{bmatrix}0_{1\times 2B-1} & 0\\ I_{2B-1} & 0\\ 0_{B\times 2B-1} & 0\end{bmatrix}Q(z) \ldots \ldots \ldots \ldots (3.2.28)$$

3.2.7 Unconstrained DHT Block Adaptive Filter Implementation

Step 1- Convert the input repressor and noise signal in blocks and then make the two consecutive input block vector into block column-wise

$$u_{B,n} \triangleq \begin{bmatrix}u(nB+B-1)\\ \vdots\\ u(nB+1)\\ u(nB)\end{bmatrix} \qquad v_{B,n} = \begin{bmatrix}v(nB+B-1)\\ \vdots\\ v(nB+1)\\ v(nB)\end{bmatrix}$$

$$u_{2B,n} = \begin{bmatrix}u_{B,n}\\ u_{B,n-1}\end{bmatrix}$$

Step 2- Now make the input block vector into transformed regressor by the DHT matrix-

$$u'_{K,n} = H\begin{bmatrix}0_{1\times 2B-1} & 0\\ I_{2B-1} & 0\\ 0_{B\times 2B-1} & 0\end{bmatrix}u_{2B,n} = col\{u'_k(n), \qquad k = 0,1,\ldots\ldots\ldots,K-1\}$$

Where $u'_k(n)$ is a transformed regressor for each sub-band filter and length of $u'_k(n)$ for each k is M/B, which is represented as-

$$u'_k(n) = \left[u'_k(n) \ldots\ldots\ldots u'_k(n-\frac{M}{B}+1)\right], \qquad k = 0,1,2,\ldots\ldots\ldots K-1$$

Step 3- The FIR filter of length B is converted into K sub-band filters, each having length M/B by the following relation (also explained above)-

$$\begin{bmatrix} L_0(z) \\ L_1(z) \\ \vdots \\ L_{K-1}(z) \end{bmatrix} = H^{-1} \begin{bmatrix} 0 \\ P_0(z) \\ \vdots \\ P_{B-1}(z) \\ 0_{B-1\times 1} \\ P_{B-1}(z) \\ \vdots \\ P_0(z) \end{bmatrix} \dots\dots\dots\dots (3.2.29)$$

Step 4- The desired output is calculated as structure of filter is depicted and converted into block vector of size B.

$$y'_k(n) = u'_{k,n} L_k, \qquad k = 0,1, \dots\dots\dots, K-1$$

$$d_{B,n} = [I_B \quad 0_{B\times 2B}] H * col\{y'_0(n), \dots\dots\dots\ y'_{K-1}(n)\} + v_{B,n}$$

Step 5- The actual output is also calculated in same fashion as-

$$y1'_k(n) = u'_{k,n} l_{k,n-1}, \qquad k = 0,1, \dots\dots\dots, K-1$$

$$\hat{d}_{B,n} = [I_B \quad 0_{B\times 2B}] H * col\{y1'_0(n), \dots\dots\dots\ y1'_{K-1}(n)\}$$

Step 6- The error vector is difference of desired output vector and actual output vector-

$$e_{B,n} = d_{B,n} - \hat{d}_{B,n}$$

Step 7- This error vector is processed by DHT matrix and last B outputs are neglected as-

$$e'_{K,n} = H \begin{bmatrix} I_B \\ 0_{2B\times B} \end{bmatrix} e_{B,n} = col\{e'_k(n), \qquad k = 0,1, \dots\dots\dots K-1\}$$

Step 8- The weight of each sub-band filter is updated separately with normalized Block LMS as-

$$l_{k,n} = l_{k,n-1} + \frac{\mu}{\lambda_k} u'^{*}_{k,n} e'_k(n), \quad k = 0,1, \dots\dots\dots (K-1) \dots\dots\dots\dots (3.2.30)$$

Where is calculated as- $\lambda_k(n) = \beta * \lambda_k(n-1) + (1-\beta)|u'_k(n)|^2$

Step 9- The actual output block vector and error block vector are interpolated with the help of size B interpolator as-

$$\hat{d}_{B,n} = col\{\hat{d}(nB+B-1), \dots\dots\dots, \hat{d}(nB+1), \hat{d}(nB)\}$$

$$e_{B,n} = col\{e(nB+B-1), \dots\dots\dots, e(nB+1), e(nB)\}$$

Step 10- Mean square error is calculated by the output of interpolator of error vector-

$$mse = e(i).* e(i)$$

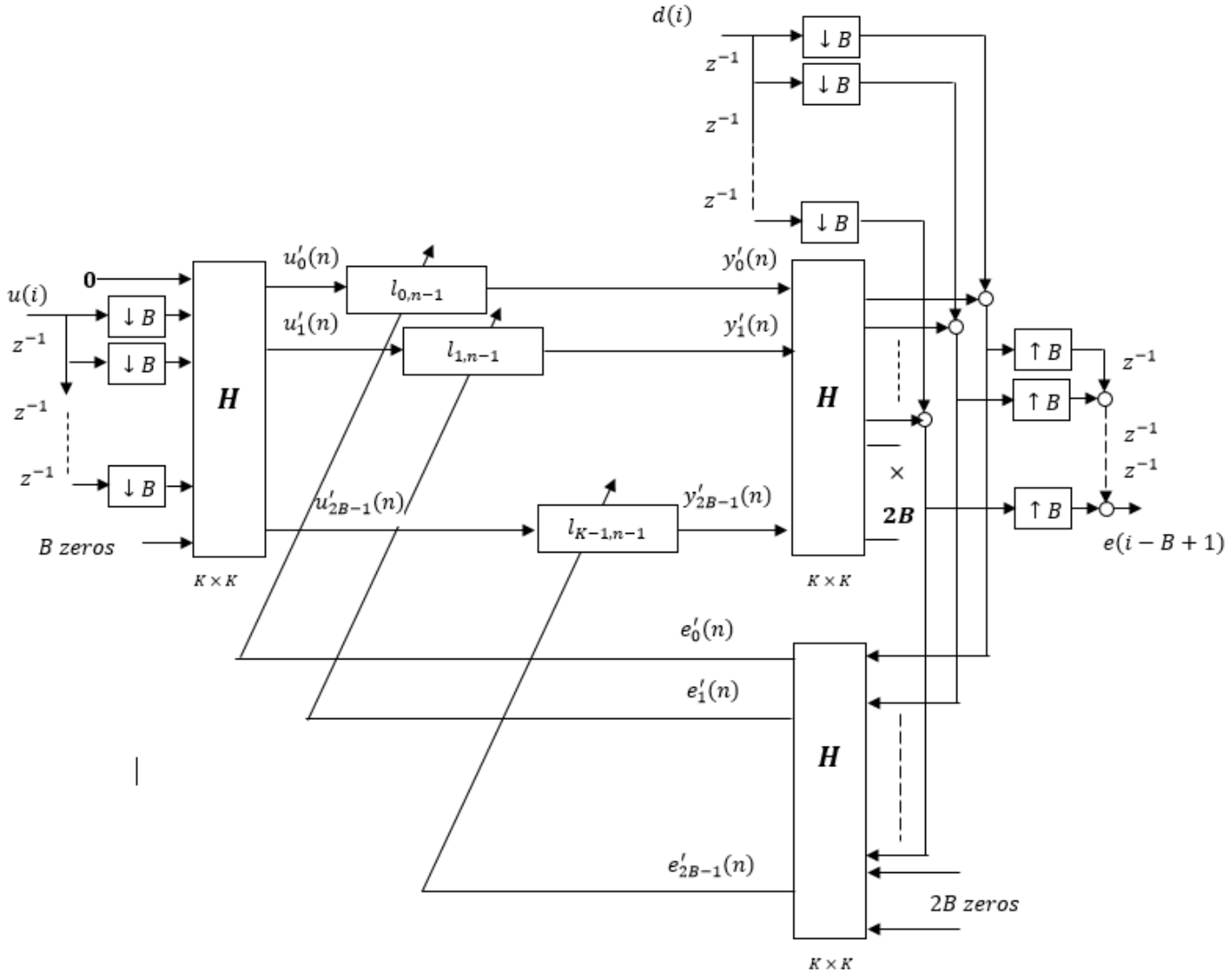


Figure-3. 12 Unconstrained DHT Block Adaptive Filter

3.2.6 Constrained DHT Block Adaptive Filter Implementation

As discussed previous in case of DFT and DCT block adaptive filter implementation the significant difference is observed between unconstrained and constrained filter implementation. The same situation repeats here in case of DHT block adaptive filter implementation.

The above unconstrained implementation is rectified at the final stage of sub-band filter weight update process. The new update equation is-

$$\begin{bmatrix} l_{0,n}^{cT} \\ l_{1,n}^{cT} \\ \vdots \\ l_{K-1}^{cT} \end{bmatrix} = H \begin{bmatrix} 0 & & & \\ & 0.5I_B & 0 & 0.5I_B^{\#} \\ & 0 & 0 & 0 \\ & 0.5I_B^{\#} & 0 & 0.5I_B \end{bmatrix} \begin{bmatrix} l_{0,n}^{T} \\ l_{1,n}^{T} \\ \vdots \\ l_{K-1}^{T} \end{bmatrix} \ldots\ \ldots\ \ldots\ \ldots \quad (3.2.31)$$

Where $I_B^{\#}$ is anti-diagonal identity matrix.

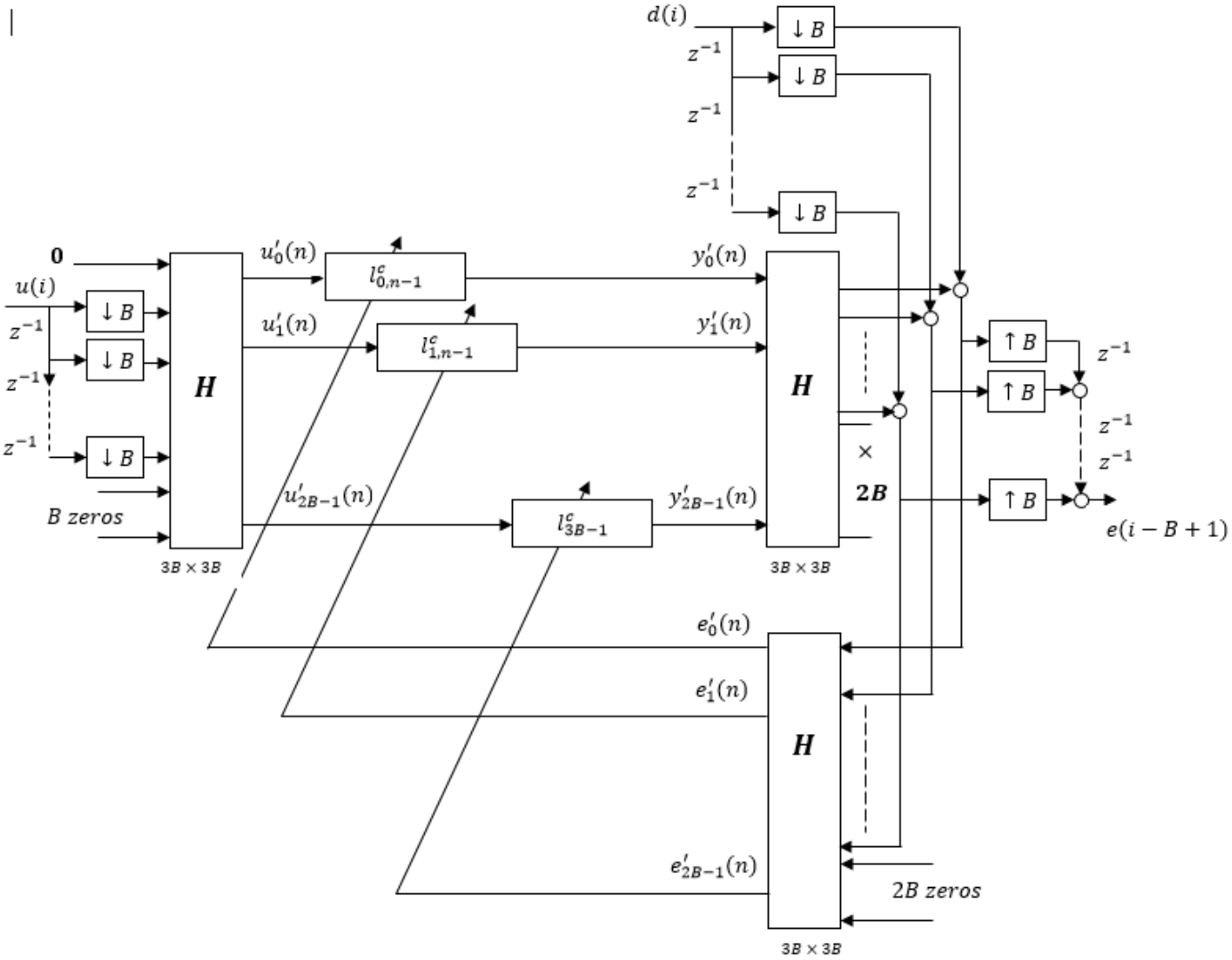


Figure-3. 13 Constrained DHT Block Adaptive Filter

3.2.7 Block Adaptive Filter Based On DCT

The block adaptive filter which based on discrete cosine transform (DCT) is motivated here. The DCT matrix of size $K \times K$ is defined as-

$$[C]_{k,m} = \alpha(k)\cos\left(\frac{k(2m+1)\pi}{2K}\right), \qquad k,m = 0,1,2,\dots\dots\dots K-1$$

Where $\alpha(0) = \frac{1}{\sqrt{K}}$ & $\alpha(k) = \sqrt{\frac{2}{K}}$, $k \neq 0$. k Denotes row index and m denotes column index. As previous discussion we know that $K \times K$ matrix $[C]_{k,m}$ diagonalizes circulant matrix $A(z)$ which can be expressed as-

$$A(z) = T(z) + H(z) + B(z) \dots \dots \dots \dots (3.2.32)$$

Where $T(z)$ is symmetric Toeplized matrix and $H(z)$ is Hankel matrix which is related to $T(z)$, $B(z)$ is border matrix which is also related to $T(z)$. For e.g. - K=4

$$T(z) = \begin{bmatrix} t_0(z) & t_1(z) & t_2(z) & t_3(z) \\ t_1(z) & t_0(z) & t_1(z) & t_2(z) \\ t_2(z) & t_1(z) & t_0(z) & t_1(z) \\ t_3(z) & t_2(z) & t_1(z) & t_0(z) \end{bmatrix} \dots \dots \dots \dots (3.2.33)$$

$$H(z) = \begin{bmatrix} t_0(z) & t_1(z) & t_2(z) & t_3(z) \\ t_1(z) & t_2(z) & t_3(z) & 0 \\ t_2(z) & t_3(z) & 0 & -t_3(z) \\ t_3(z) & 0 & -t_3(z) & -t_2(z) \end{bmatrix} \dots \dots \dots \dots (3.2.34)$$

$$B(z) = \begin{bmatrix} -\frac{t_0(z)}{\sqrt{2}} - 1 & t_1(z) & t_2(z) & t_3(z) \\ t_1(z) & 0 & 0 & 0 \\ t_2(z) & 0 & 0 & 0 \\ t_3(z) & 0 & 0 & 0 \end{bmatrix} \left(\sqrt{2} - 2\right) \dots \dots \dots \dots (3.2.35)$$

For B=2

$$P(z) = \begin{bmatrix} P_0(z) & P_1(z) & 0 \\ 0 & P_0(z) & P_1(z) \end{bmatrix} \dots \dots \dots \dots (3.2.36)$$

$$T(z) = \begin{bmatrix} 0 & P_0(z) & P_0(z) & 0 & 0 \\ P_0(z) & 0 & P_0(z) & P_0(z) & 0 \\ P_0(z) & P_0(z) & 0 & P_0(z) & P_0(z) \\ 0 & P_0(z) & P_0(z) & 0 & P_0(z) \\ 0 & 0 & P_0(z) & P_0(z) & 0 \end{bmatrix} \ldots\ \ldots\ \ldots\ \ldots\ (3.2.37)$$

$$A(z) = \begin{bmatrix} 0 & \sqrt{2}P_0(z) & \sqrt{2}P_0(z) & 0 & 0 \\ \sqrt{2}P_0(z) & 0 & P_0(z) & P_0(z) & 0 \\ \sqrt{2}P_0(z) & P_0(z) & 0 & P_0(z) & P_0(z) \\ 0 & P_0(z) & P_0(z) & 0 & P_0(z) \\ 0 & 0 & P_0(z) & P_0(z) & 0 \end{bmatrix} \ldots\ \ldots\ \ldots\ \ldots\ (3.2.38)$$

The dimension of $K \times K$ matrix is defined ass-

$$K = \begin{cases} \frac{7B-4}{2} & if\ B\ is\ even \\ \frac{7B-3}{2} & if\ B\ is\ odd \end{cases} \ldots\ \ldots\ \ldots\ \ldots\ (3.2.39)$$

For any one block of size B, the first row of $T(z)$ is having the formation like-

$$[0_{1\times B-1} \quad P_0(z) \ldots P_{B-1}(z) \quad 0_{1\times\alpha}]$$

Which is having (B-1) no. of zeros in starting and α no. of zeros in ending.

Where α is defined as-

$$\alpha = \begin{cases} \frac{3B-2}{2} & if\ B\ is\ even \\ \frac{3B-1}{2} & if\ B\ is\ odd \end{cases} \ldots\ \ldots\ \ldots\ \ldots\ (3.2.40)$$

Then $P(z)$ is derived from $A(z)$ as-

$$P(z) = \left[0_{B\times\gamma} \quad I_B \quad 0_{B\times 2B-2}\right] A(z) \begin{bmatrix} 0_{\alpha\times(2B-1)} \\ I_{2B-1} \end{bmatrix} \ldots\ \ldots\ \ldots\ \ldots\ (3.2.41)$$

Where γ is defined as-

$$\gamma = \begin{cases} \frac{B}{2} & if\ B\ is\ even \\ \frac{B+1}{2} & if\ B\ is\ odd \end{cases} \ldots\ \ldots\ \ldots\ \ldots\ (3.2.42)$$

The matrix $A(z)$ can be diagonalized by C as-

$$A(z) = CL(z)C^T \dots \dots \dots \dots (3.2.43)$$

Where $L(z) = diag\{L_k(z)\}$ has K entries. The relation is given as-

$$\sqrt{2K}\begin{bmatrix} 0_{(B-1)\times 1} \\ P_0(z) \\ \vdots \\ P_{B-1}(z) \\ 0_{\alpha\times 1} \end{bmatrix} = C\begin{bmatrix} L_0(z) \\ L_1(z) \\ \vdots \\ L_{K-1}(z) \end{bmatrix} \dots \dots \dots \dots (3.2.44)$$

Now the final decomposition is –

$$G'(z) = \begin{bmatrix} 0_{B\times\gamma} & I_B & 0_{B\times 2B-2} \end{bmatrix} A(z) \begin{bmatrix} 0_{\alpha\times(2B-1)} \\ I_{2B-1} \end{bmatrix} Q(z)$$

$$G'(z) = \begin{bmatrix} 0_{B\times\gamma} & I_B & 0_{B\times 2B-2} \end{bmatrix} CL(z)C^T \begin{bmatrix} 0_{\alpha\times(2B-1)} \\ I_{2B-1} \end{bmatrix} Q(z)$$

$$G'(z) = \begin{bmatrix} 0_{B\times\gamma} & I_B & 0_{B\times 2B-2} \end{bmatrix} CL(z)C^T \begin{bmatrix} 0_{\alpha\times(2B-1)} & 0 \\ I_{2B-1} & 0 \end{bmatrix} Q(z) \dots \dots \dots \dots (3.2.45)$$

3.2.8 Unconstrained DCT Block Adaptive Filter Implementation

As discussed above the block adaptive implementation requires block-by-block processing with transform domain implementation of data. Both of these combined technique give better computational cost reduction as well as better convergence performance. Let's make the analysis step-by-step;-

Step 1- Convert the input repressor and noise signal in blocks and then make the two consecutive input block vector into block column-wise

$$u_{B,n} \triangleq \begin{bmatrix} u(nB+B-1) \\ \vdots \\ u(nB+1) \\ u(nB) \end{bmatrix} \qquad v_{B,n} = \begin{bmatrix} v(nB+B-1) \\ \vdots \\ v(nB+1) \\ v(nB) \end{bmatrix}$$

$$u_{2B,n} = \begin{bmatrix} u_{B,n} \\ u_{B,n-1} \end{bmatrix}$$

Step 2- Now make the input block vector into transformed regressor by the DCT matrix-

$$u'_{K,n} = C^T \begin{bmatrix} 0_{\alpha\times(2B-1)} & 0 \\ I_{2B-1} & 0 \end{bmatrix} u_{2B,n} = col\{u'_k(n), \qquad k = 0,1, \dots \dots \dots, K-1\}$$

Where $u'_k(n)$ is a transformed regressor for each sub-band filter and length of $u'_k(n)$ for each k is M/B, which is represented as-

$$u'_k(n) = \left[u'_k(n) \dots \dots \dots \; u'_k(n - \frac{M}{B} + 1)\right], \qquad k = 0,1,2, \dots \dots \dots K-1$$

Step 3- The FIR filter of length B is converted into K sub-band filters, each having length M/B by the following relation (also explained above)-

$$\begin{bmatrix} L_0(z) \\ L_1(z) \\ \vdots \\ L_{K-1}(z) \end{bmatrix} = \sqrt{2K}C^{-1} \begin{bmatrix} 0_{(B-1)\times 1} \\ P_0(z) \\ \vdots \\ P_{B-1}(z) \\ 0_{\alpha \times 1} \end{bmatrix}$$

Step 4- The desired output is calculated as structure of filter is depicted and converted into block vector of size B.

$$y'_k(n) = u'_{k,n}L_k, \qquad k = 0,1, \dots \dots \dots, K-1$$

$$d_{B,n} = \begin{bmatrix} 0_{B\times\gamma} & I_B & 0 \end{bmatrix} C * col\{y'_0(n), \dots \dots y'_{K-1}(n)\} + v_{B,n}$$

Step 5- The actual output is also calculated in same fashion as-

$$y1'_k(n) = u'_{k,n}l_{k,n-1}, \qquad k = 0,1, \dots \dots \dots, 2B-1$$

$$\hat{d}_{B,n} = \begin{bmatrix} 0_{B\times\gamma} & I_B & 0 \end{bmatrix} C * col\{y'_0(n), \dots \dots y1'_{K-1}(n)\}$$

Step 6- The error vector is difference of desired output vector and actual output vector-

$$e_{B,n} = d_{B,n} - \hat{d}_{B,n}$$

Step 7- This error vector is processed by DFT matrix and last B outputs are neglected as-

$$e'_{2B,n} = F \begin{bmatrix} I_B \\ 0_{B\times B} \end{bmatrix} e_{B,n} = col\{e'_k(n), \qquad k = 0,1, \dots \dots \dots K-1\}$$

Step 8- The weight of each sub-band filter is updated separately with normalized Block LMS as-

$$l_{k,n} = l_{k,n-1} + \frac{\mu}{\lambda_k} u'^{*}_{k,n} e'_k(n), \qquad k = 0,1, \dots (K-1) \; \dots \; \dots \; \dots \; \dots \; (3.2.46)$$

Where is calculated as- $\lambda_k(n) = \beta\lambda_k(n-1) + (1-\beta)|u'_k(n)|^2$

Step 9- The actual output block vector and error block vector are interpolated with the help of size B interpolator as-

$$\hat{d}_{B,n} = col\{\hat{d}(nB + B - 1), \dots\dots\dots, \hat{d}(nB + 1), \hat{d}(nB)\}$$

$$e_{B,n} = col\{e(nB + B - 1), \dots\dots\dots, e(nB + 1), e(nB)\}$$

Step 10- Mean square error is calculated by the output of interpolator of error vector-

$$mse = e(i).* e(i)$$

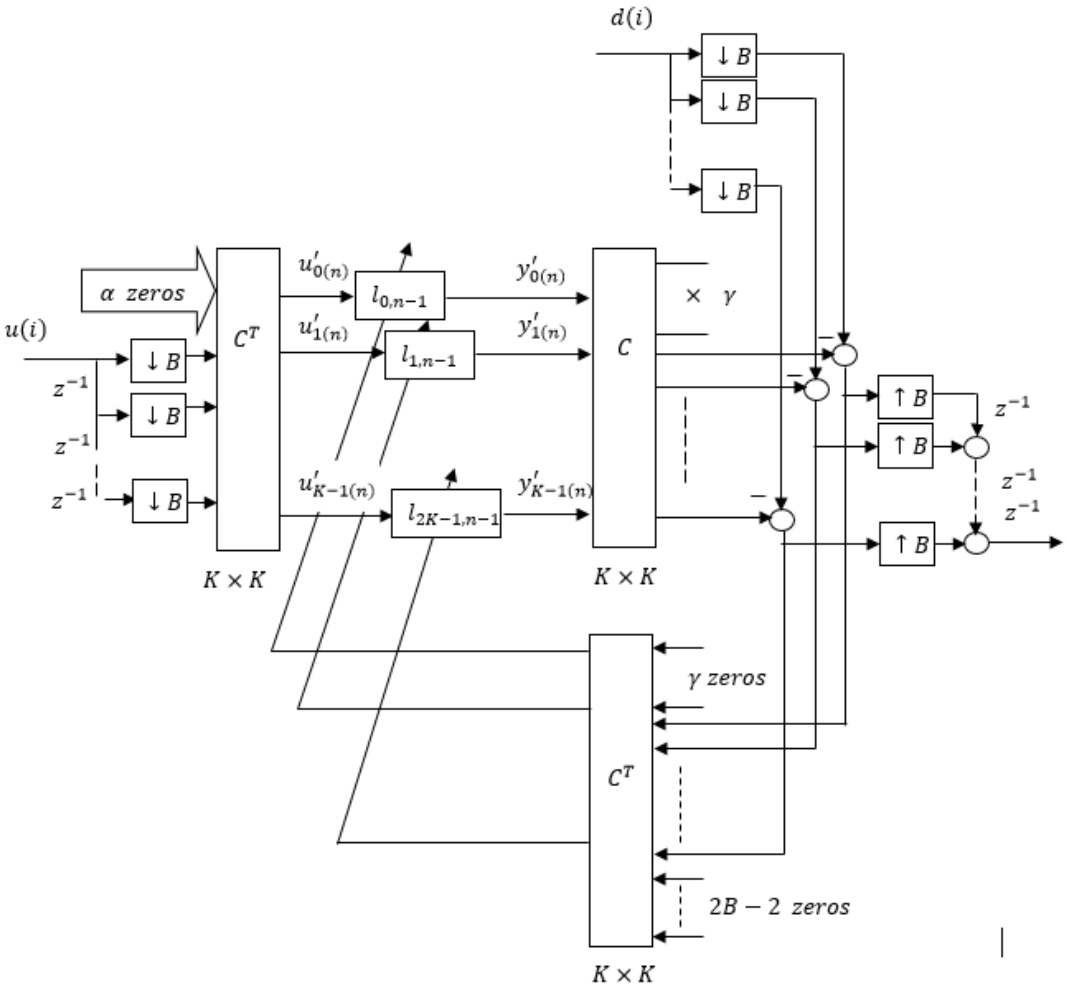


Figure-3. 14 Unconstrained DCT Block Adaptive Filter

3.2.9 Constrained DCT Block Adaptive Filter Implementation

As discussed previous in case of DFT block adaptive filter implementation the significant difference is observed between unconstrained and constrained filter implementation. The same situation repeats here in case of DCT block adaptive filter implementation.

The above unconstrained implementation is rectified at the final stage of sub-band filter weight update process. The new update equation is-

$$\begin{bmatrix} l_{0,n}^{cT} \\ l_{1,n}^{cT} \\ \vdots \\ l_{K-1}^{cT} \end{bmatrix} = C^T \begin{bmatrix} 0_{B\times(B-1)} & I_B & 0_{B\times\alpha} \end{bmatrix} C \begin{bmatrix} l_{0,n}^{T} \\ l_{1,n}^{T} \\ \vdots \\ l_{K-1}^{T} \end{bmatrix} \ldots \ldots \ldots \ldots (3.2.47)$$

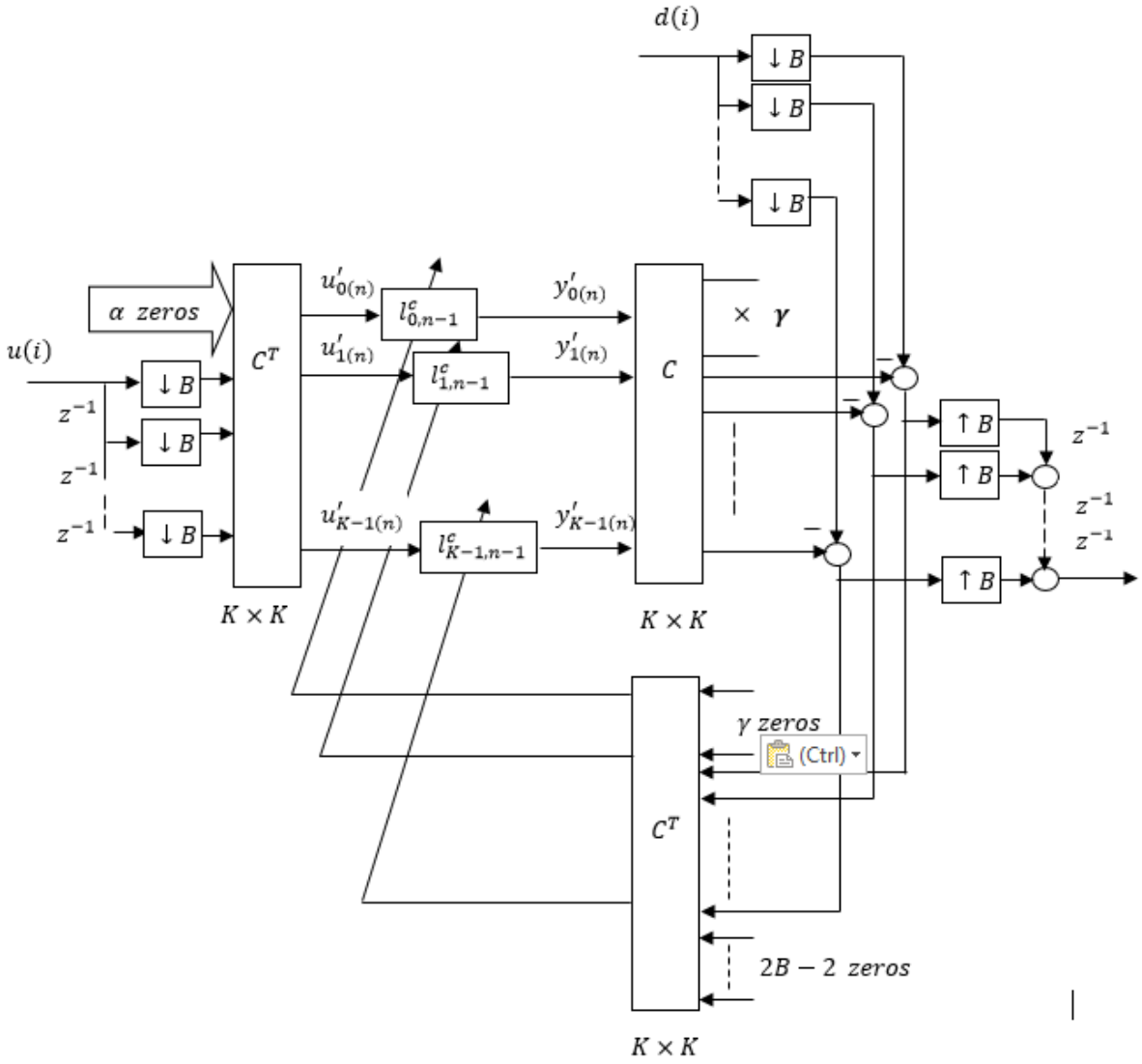


Figure-3. 15 Constrained DCT Block Adaptive Filter

3.2.10 Computational complexity of overall block adaptive filter

The computational complexity of all block adaptive filter can be calculated very easily. We are computing here computational cost for constrained DFT block adaptive filter. The entire process of cost evaluation is divided into four stages.

1. **Decomposition of sub-bands of input and error signal**

   The block size is B for input and error signal $\{u(i), e(i)\}$ and their transformed vectors $\{u'_{2B,n}, e'_{2B,n}\}$ respectively. For each block of B input samples, DFT of size 2B is needed. We know that cost of A size DFT is $\frac{A}{2}\log_2 A$ complex operations.

$$\frac{1}{B}2(B\log_2(B)) = 2\log_2 2B \quad operations\ per\ unit\ sample$$

2. **Updating process of sub-band filters**

   There are 2B sub-band filters each having length M/B by using N-LMS with power normalization. The update is for one input block size B. P-long N-LMS filter needed about 2P complex operations. So-

$$\frac{1}{B}2B\frac{2M}{B} = \frac{4M}{B} \qquad operations\ per\ unit\ sample$$

3. **Enforcement of constraints**

   The step in which sub-band filter weights $L_k(z)$ are processed with DFT matrix, total 2B sub-band filters with each of size M/B. the complexity is very similar to part one but here we need to compute for M/B transforms.

$$\frac{1}{B} * \frac{M}{B}(B\log_2(B)) = M\log_2 2B \qquad operations\ per\ unit\ sample$$

4. **Inverse transformation**

   The mapping of signal from $y'_k(n)$ into $y_k(n)$ needed one DFT of size 2B.

$$B\log_2(2B) \qquad operations\ per\ unit\ sample \quad \ldots\ \ldots\ \ldots\ \ldots \quad (3.2.48)$$

Finally we can say that

$$\frac{4M}{B} + \left(\frac{M}{B} + 3\right)\log_2(2B) \qquad operations\ per\ unit\ sample$$

The over-all conclusion is that the computational cost is $O(M/B)$ operations per sample where as in simple FIR filter it is $O(M)$ operations per samples.

CHAPTER 4

FREQUENCY DOMAIN FILTERING BY UNITARY TRANSFORMS IN DIFFUSION & INCREMENTAL STRATEGIES OVER DISTRIBUTED NETWORKS

By the help of unitary transforms like DCT & DFT we can improve the convergence performance of an adaptive filter. The unitary transforms DFT & DCT have orthogonal properties, which are helpful in input data de-correlation. Hence the eigen value spread of covariance matrix of input data is less and we can easily get better convergence performance. Two basic steps must be followed in this implementation-

1> The input regressor must be processed by transform matrix at each node.

2> Power normalization of transformed input regressor at each node.

Let's s denotes the no. of nodes ($s = 1,2, \dots \dots N$). N denotes total no. of nodes.

The weight updating equation of Adaptive Filter is-

$$W_{s,i} = W_{s,i-1} + \mu u'_{s,i}\left(d_s(i) - u_{s,i}W_{s,i-1}\right), \qquad s = 1,2, \dots \dots N$$

The length of filter is considered as M. So the size unitary transform matrix is $M \times M$.

The DFT matrix is defined as-

$$[F]_{km} = \frac{1}{\sqrt{M}} e^{-\frac{j2\pi mk}{M}}, \qquad k, m = 0,1,2, \dots \dots \dots M-1$$

Similarly the DCT matrix is defined as-

$$[C]_{km} = \alpha(k)\cos\left(\frac{k(2m+1)\pi}{2M}\right), \qquad k, m = 0,1,2, \dots \dots \dots M-1$$

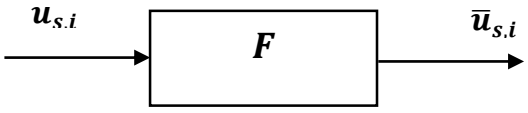
Where

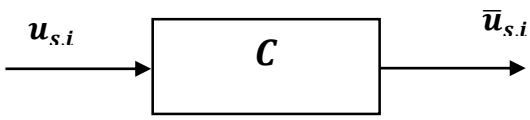
$$\alpha(0) = \frac{1}{\sqrt{M}} \qquad and \qquad \alpha(k) = \frac{2}{\sqrt{M}} \quad for\ k \neq 0$$

The transformed regressor is –

$$\bar{u}_{s,i} = u_{s,i}T$$

Which give the transformed regressor in DFT & DCT as-





Now the weight matrix of filter is also processed by unitary transform matrix as-

$$\bar{W}_{s,i} = T^* W_{s,i}, \qquad s = 1,2, \dots \dots N$$

Now the weight updating equation can be written as-

$$\bar{W}_{s,i} = \bar{W}_{s,i-1} + \mu \bar{u}^*_{s,i}\big(d_s(i) - \bar{u}_{s,i} W_{s,i-1}\big), \quad \bar{W}_{s,-1} = 0, \ \ s = 1, \dots N \qquad \dots (4.1)$$

Now in order to proceed for power normalization process, means the input regressor is divide by input power at each node to normalize.

Let's define a new term –

$$\lambda_{s,k}(i) = \beta \lambda_{s,k}(i-1) + (1-\beta)\left|\bar{u}_{s,i}(k)\right|^2, \quad k = 0,1,2, \dots M-1, s = 1,2, \dots N \dots (4.2)$$

Where $0 \ll \beta < 1$.generally β is very close to one $\bar{u}_i(k)$ denotes the k-th entry of regressor $\bar{u}_i$

With the help of this power normalization factor, a diagonal matrix D is defined as –

$$D_{s,i} = diag\{\lambda_{s,k}(i)\} \dots \dots (4.3)$$

Finally including all required concept, the weight updating equation becomes-

$$\bar{W}_{s,i} = \bar{W}_{s,i-1} + \mu D^{-1}_{s,i} \bar{u}^*_{s,i} e_s(i) \dots \dots (4.4)$$

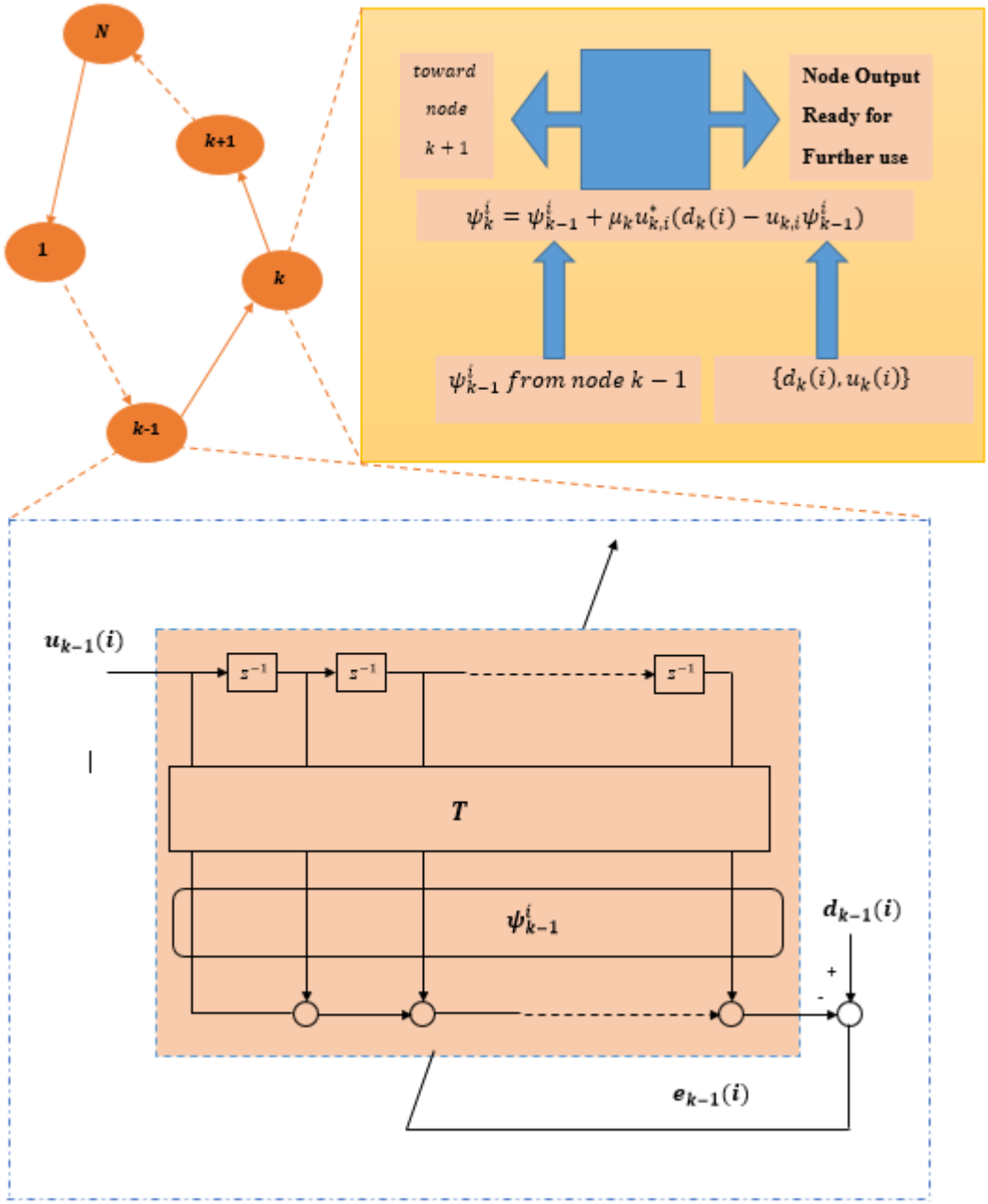


Figure 4. 1 Working of nodes in Incremental strategy by Transform Domain Filtering

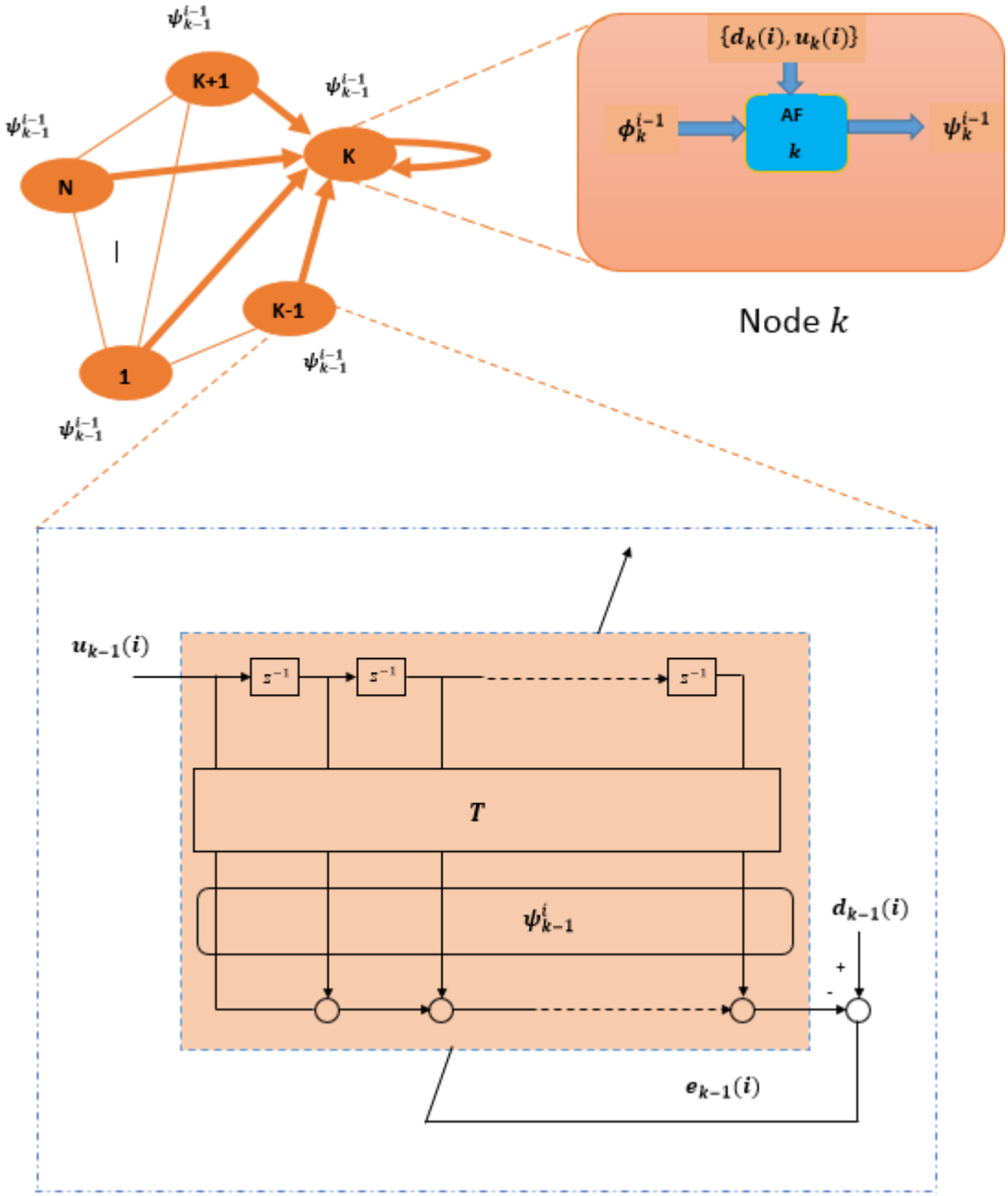


Figure 4. 2 Working of nodes in Diffusion strategy by Transform Domain Filtering

4.1 Block adaptive filtering in diffusion strategy over distributed network

A distributed network containing many nodes and nodes are connected to each other by diffusion co-operative scheme. The purpose of each node is to estimate desire parameter of interest by exchanging information with pre-defined neighbouring nodes. Each node is considered as an adaptive filter. As many of block adaptive filters are discussed above, the important thing is that, each filter is further divided into many sub filters. The number of sub-filter of a particular adaptive filter is depends on the size of block B.

We can consider a particular node as a group of sub-node. Suppose a node is exchanging data with other node, it means all the sub-nodes of that particular node are exchanging information with all sub-nodes of other node. Each sub-node can exchanging the information with other sub-node at equal position. Foe e.g. node A and B are sharing information to each other, then first sub-node of node A can exchange information with first sub-node of node B. For e.g. 4 nodes are in a network and connected to each other and each node is having 4 sub nodes as shown-

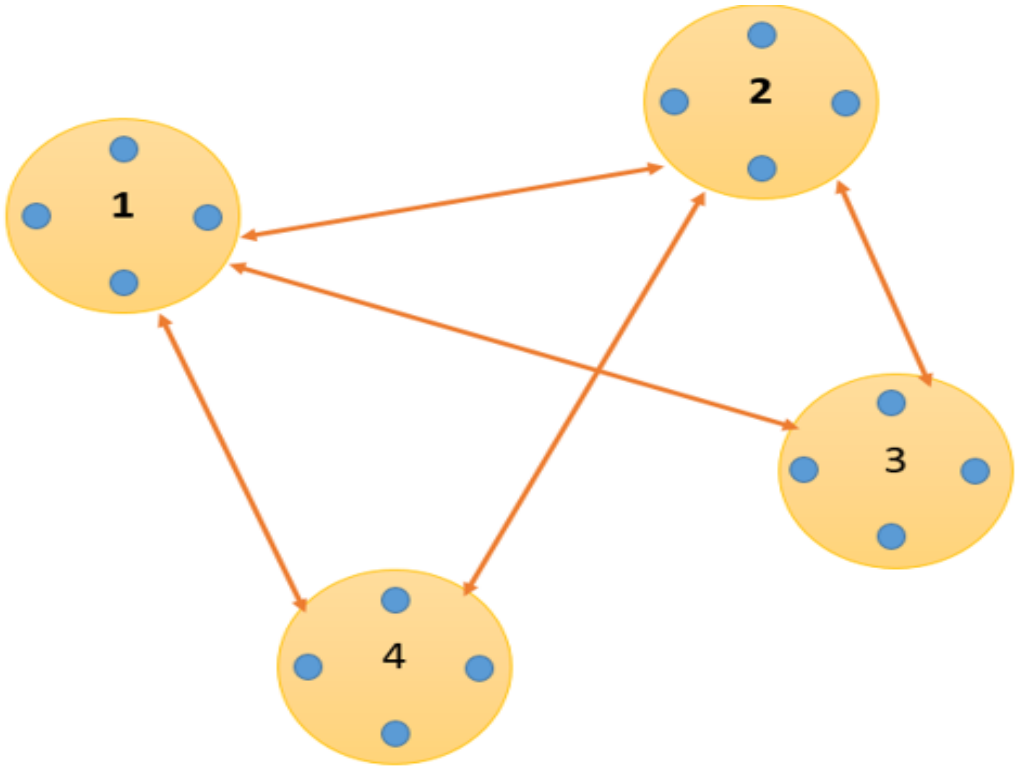


Figure 4. 3 Nodes with four sub nodes

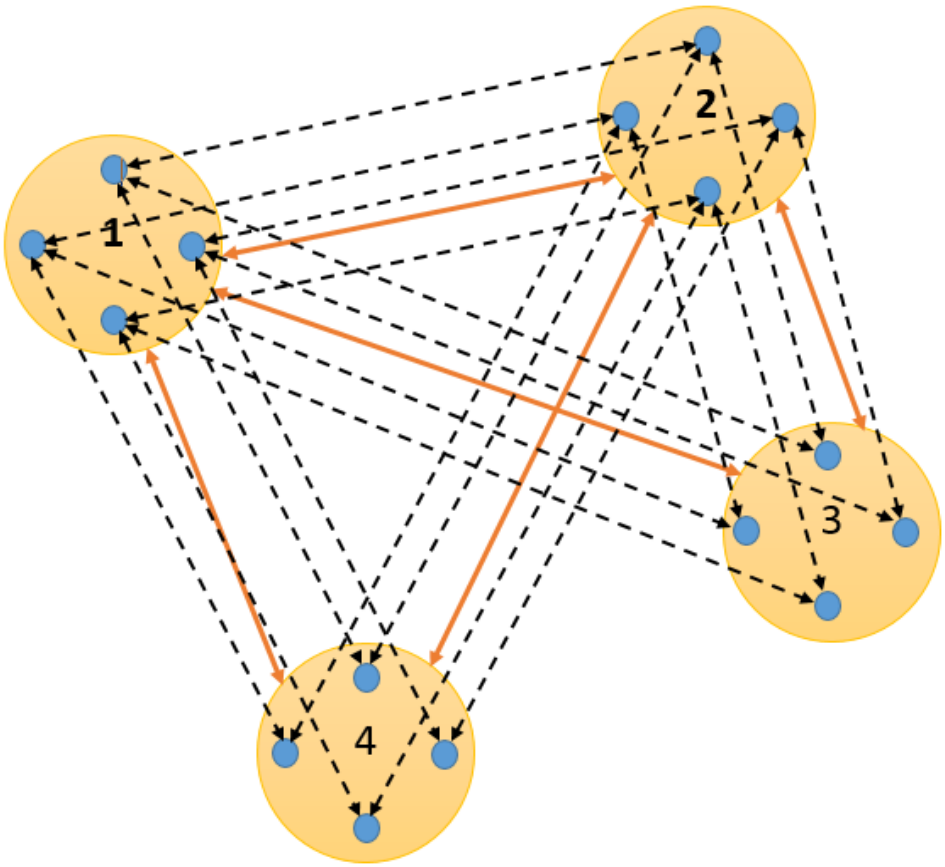


Figure 4. 4 The connectivity between sub-nodes of each node

4.2 DFT block adaptive filtering Processing in diffusion co-operative scheme

1> Block Adaptive filter based on DFT :-

Each node is considered here as a block adaptive filter, which is consisting of many sub-nodes. The sub-nodes are represented by k. total no of modes are N and node is represented by s.

Step 1- Convert the input repressor and noise signal of each node in blocks and then make the two consecutive input block vector into block column-wise

$$u_{s,B,n} \triangleq \begin{bmatrix} u_{s(nB+B-1)} \\ \vdots \\ u_{s(nB+1)} \\ u_{s(nB)} \end{bmatrix} \qquad v_{s,B,n} = \begin{bmatrix} v_{s(nB+B-1)} \\ \vdots \\ v_{s(nB+1)} \\ v_{s(nB)} \end{bmatrix}$$

$$u_{s,2B,n} = \begin{bmatrix} u_{s,B,n} \\ u_{s,B,n-1} \end{bmatrix}$$

Step 2- Now make the input block vector into transformed regressor at each node s by the DFT matrix-

$$u'_{s,2B,n} = F * u_{s,2B,n} = col\{u'_{s,k}(n), \qquad s = 1,2, \dots N, \qquad k = 0,1,2B-1\}$$

Where $u'_{s,k}(n)$a transformed regressor is for each node s and each sub-band filter k and length of $u'_{s,k}(n)$ is M/B for each k, which is represented as-

$$u'_{s,k}(n) = \left[u'_k(n) \dots \dots \dots \; u'_{s,k}(n - \frac{M}{B} + 1)\right], \qquad s = 1,2, \dots N, \qquad k = 0,1, \dots 2B-1$$

Step 3- The FIR filter of length M is converted into 2B sub-band filters, each having length M/B by the following relation (also explained above)-

$$\begin{bmatrix} L_{s,0}(z) \\ L_{s,1}(z) \\ \vdots \\ L_{s,2B-1}(z) \end{bmatrix} = \frac{1}{2B} F^* \begin{bmatrix} P_{s,0}(z) \\ \vdots \\ P_{s,B-1}(z) \\ 0_{B\times 1} \end{bmatrix} \dots \dots \; (4.5)$$

Step 4- The desired output is calculated as structure of filter is depicted and converted into block vector of size B.

$$y'_{s,k}(n) = u'_{s,k,n} L_{s,k}, \qquad k = 0,1, \dots \dots \dots, 2B-1 \dots \dots \; (4.6)$$

$$d_{s,B,n} = [I_B \quad 0_{B\times B}] F^* y'_{s,k}(n) + v_{s,B,n} \dots \dots \; (4.7)$$

Step 5- The actual output is also calculated in same fashion as-

$$y1'_{s,k}(n) = u'_{s,k,n} l_{s,k,n-1}, \qquad k = 0,1, \dots \dots \dots, 2B-1$$

$$\hat{d}_{s,B,n} = [I_B \quad 0_{B\times B}] F^* y1'_{s,k}(n) \dots \dots \; (4.8)$$

Step 6- The error vector is difference of desired output vector and actual output vector-

$$e_{s,B,n} = d_{s,B,n} - \hat{d}_{s,B,n}$$

Step 7- This error vector is processed by DFT matrix and last B outputs are neglected as-

$$e'_{s,2B,n} = F\begin{bmatrix} I_B \\ 0_{B\times B} \end{bmatrix} e_{s,B,n} = col\{e'_{s,k}(n), \qquad k = 0,1, \dots \dots \dots 2B-1\}$$

Step 8- The weight of each sub-band filter is updated separately with normalized Block LMS as-

$$l_{s,k,n} = l_{s,k,n-1} + \frac{\mu}{\lambda_k} u'^{*}_{s,k,n} e'_{s,k}(n), \qquad k = 0,1, \dots 2B-1 \ \ \& \ \ s = 1,2, \dots N \dots \dots \quad (4.9)$$

Where $\lambda_{s,k}$ is calculated as-

$$\lambda_{s,k}(n) = \beta\lambda_{s,k}(n-1) + (1-\beta)\left|u'_{s,k}(n)\right|^2, \qquad s = 1,2, \dots N \qquad \dots \ \dots \ (4.10)$$

Step 9- The actual output block vector and error block vector are interpolated with the help of size B interpolator as-

$$\hat{d}_{s,B,n} = col\{\hat{d}_s(nB+B-1), \dots \dots \dots, \hat{d}_s(nB+1), \hat{d}_s(nB)\}$$

$$e_{s,B,n} = col\{e_s(nB+B-1), \dots \dots \dots, e_s(nB+1), e_s(nB)\}$$

CHAPTER 5
RESULTS

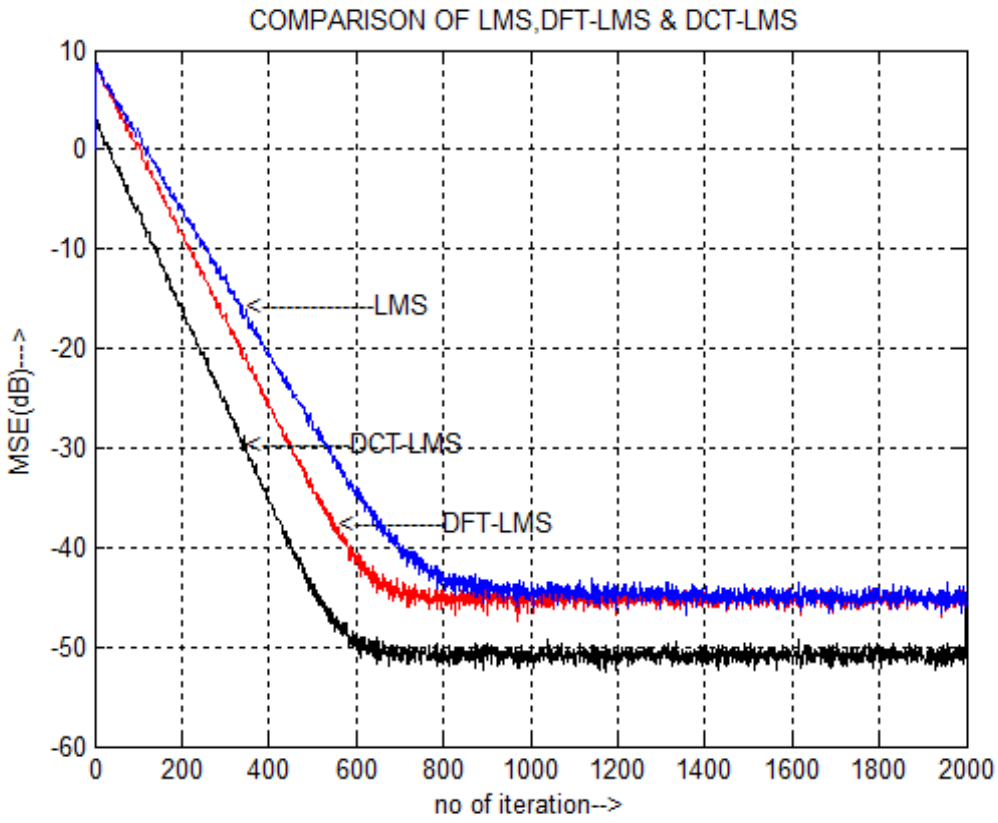


Figure-5. 1 Comparison of LMS, DFT-LMS & DCT-LMS

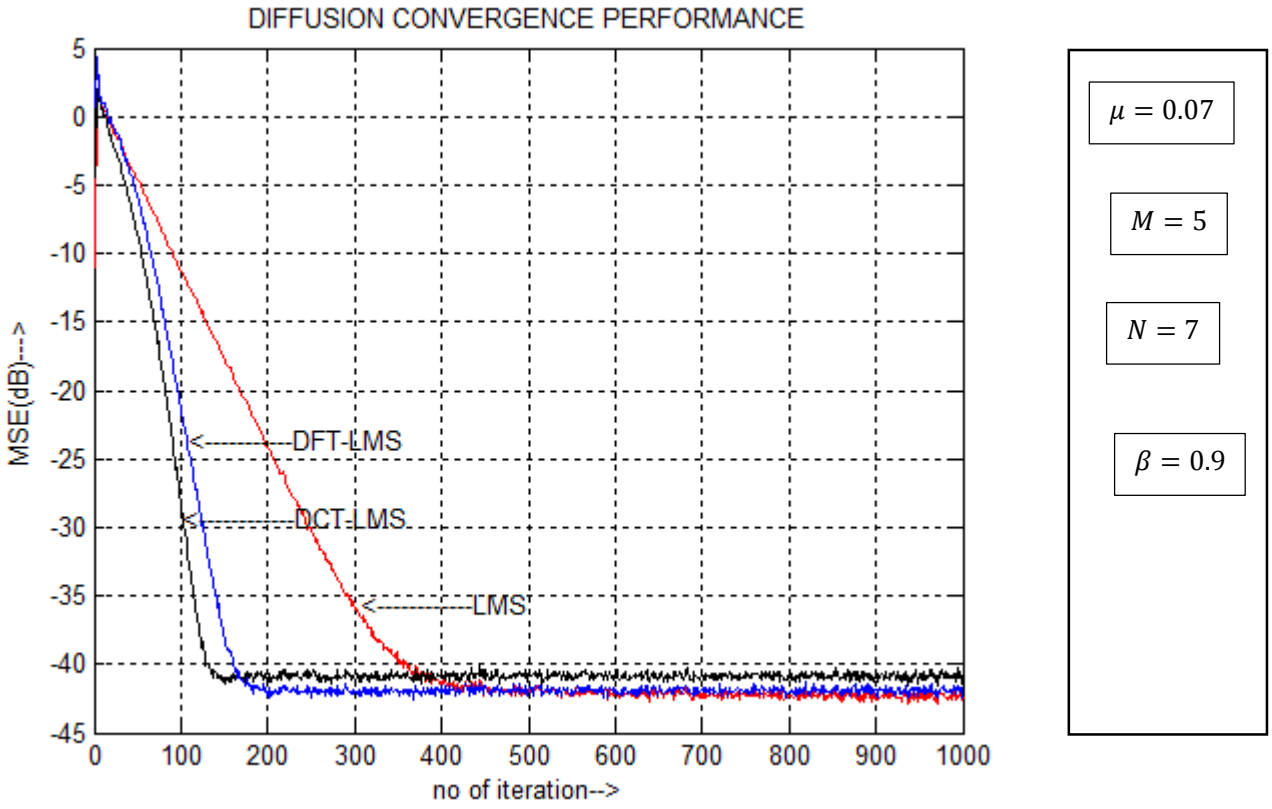


Figure-5. 2 Diffusion convergence performance

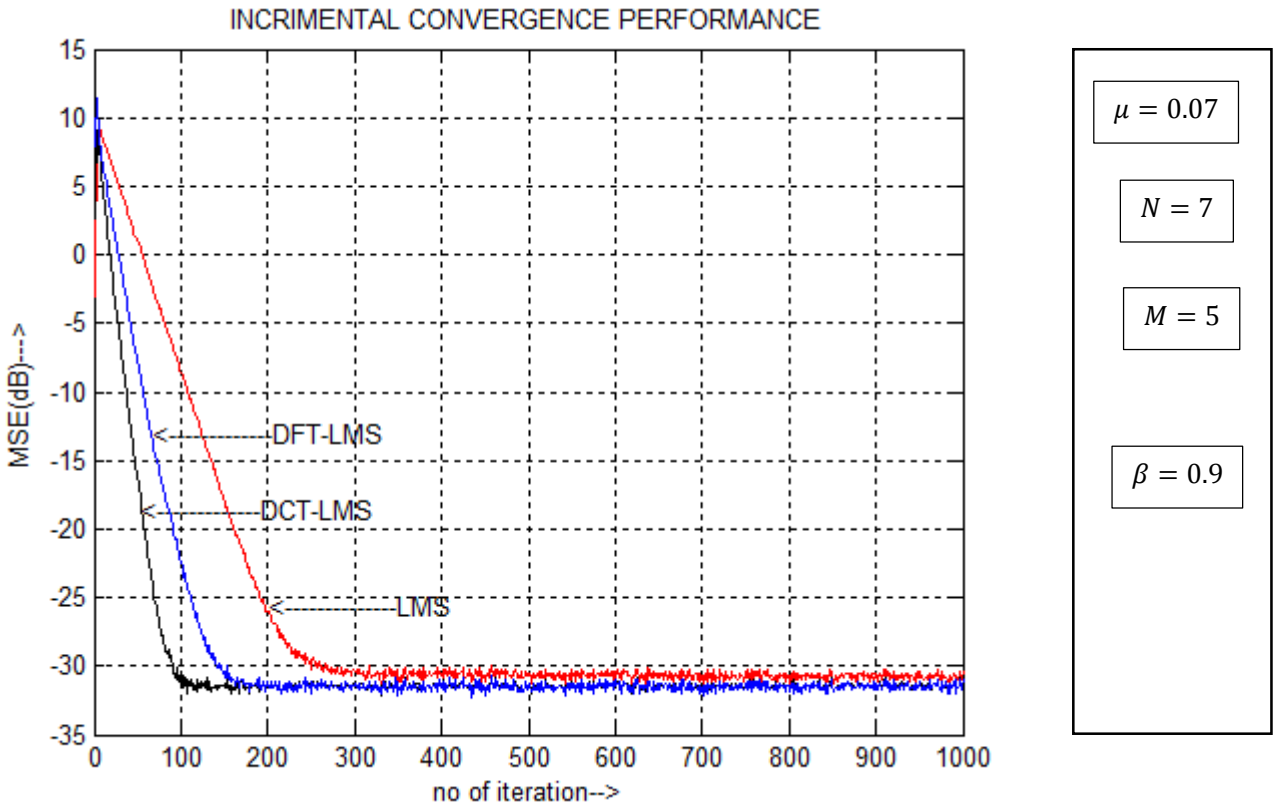


Figure-5. 3 Incremental convergence performance

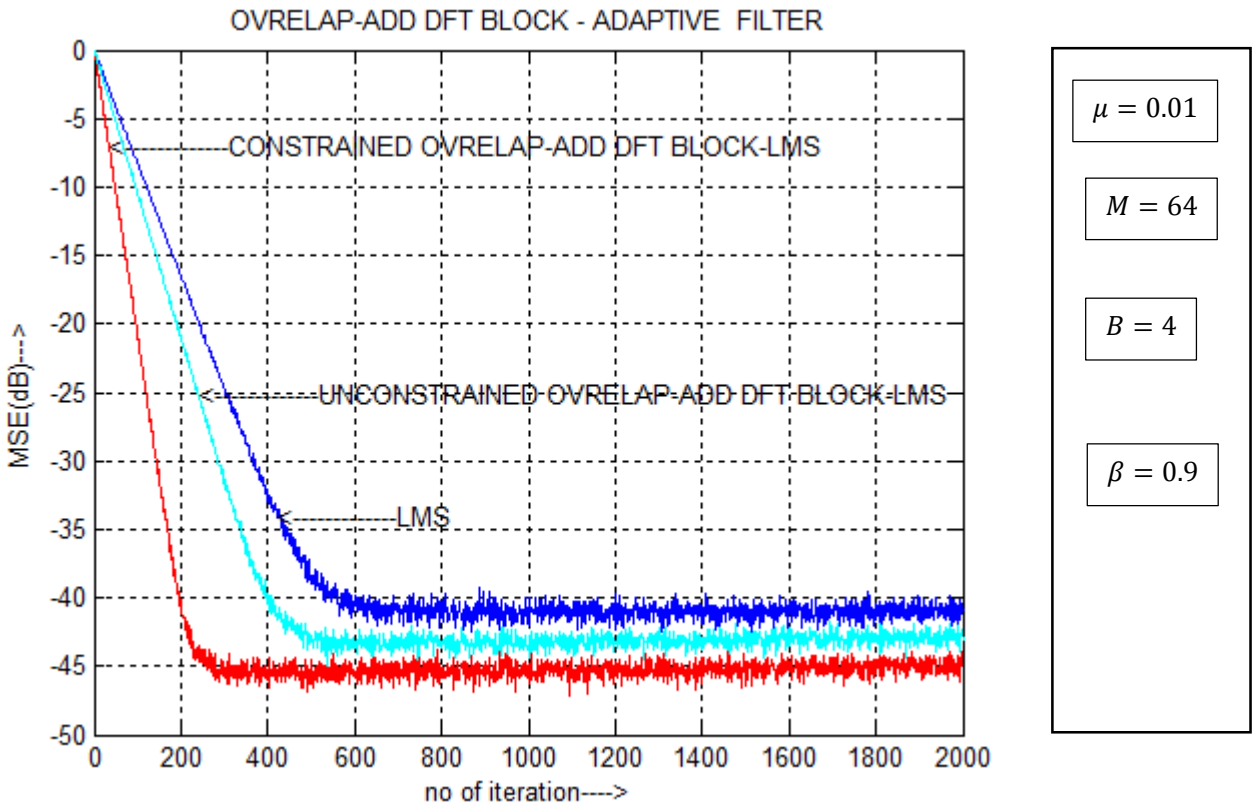


Figure-5. 4 Overlap-Add DFT Block Adaptive Filter

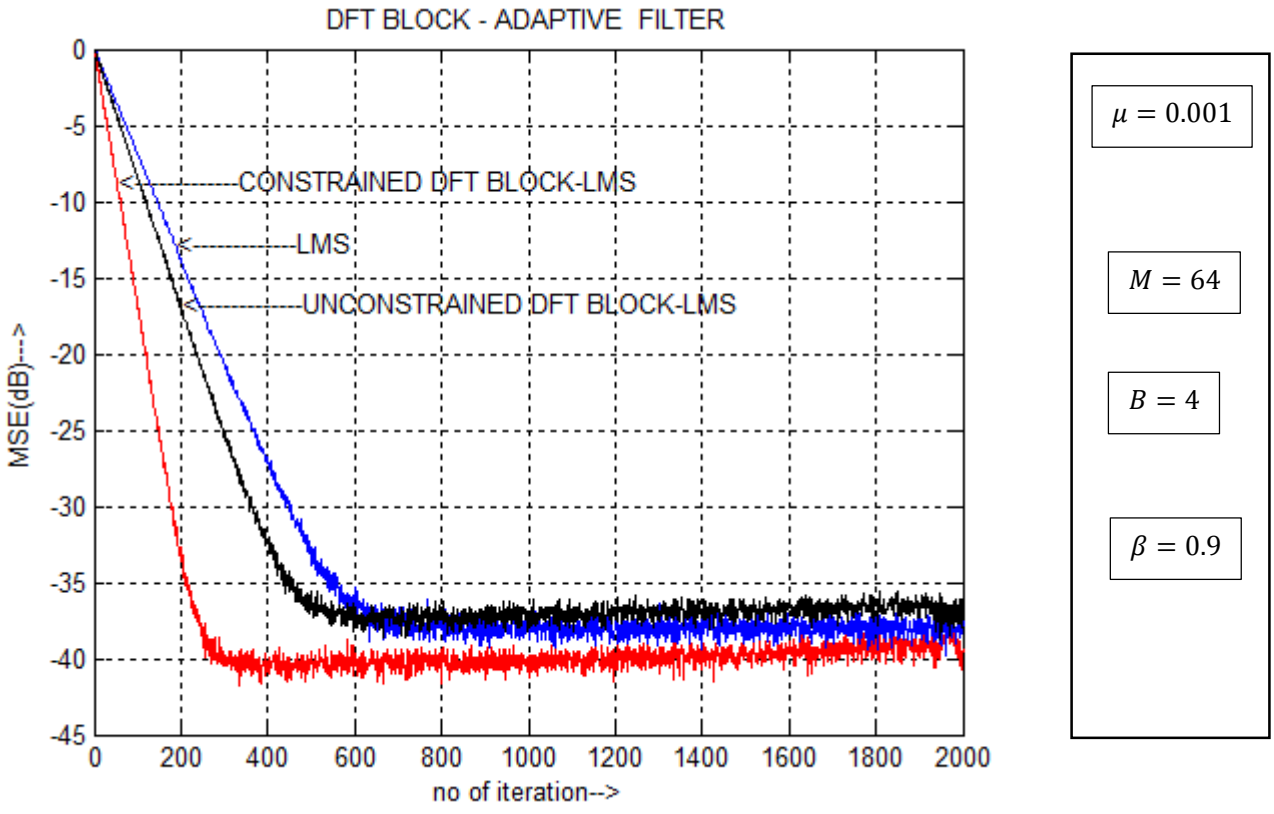


Figure-5. 5 DFT Block Adaptive Filter

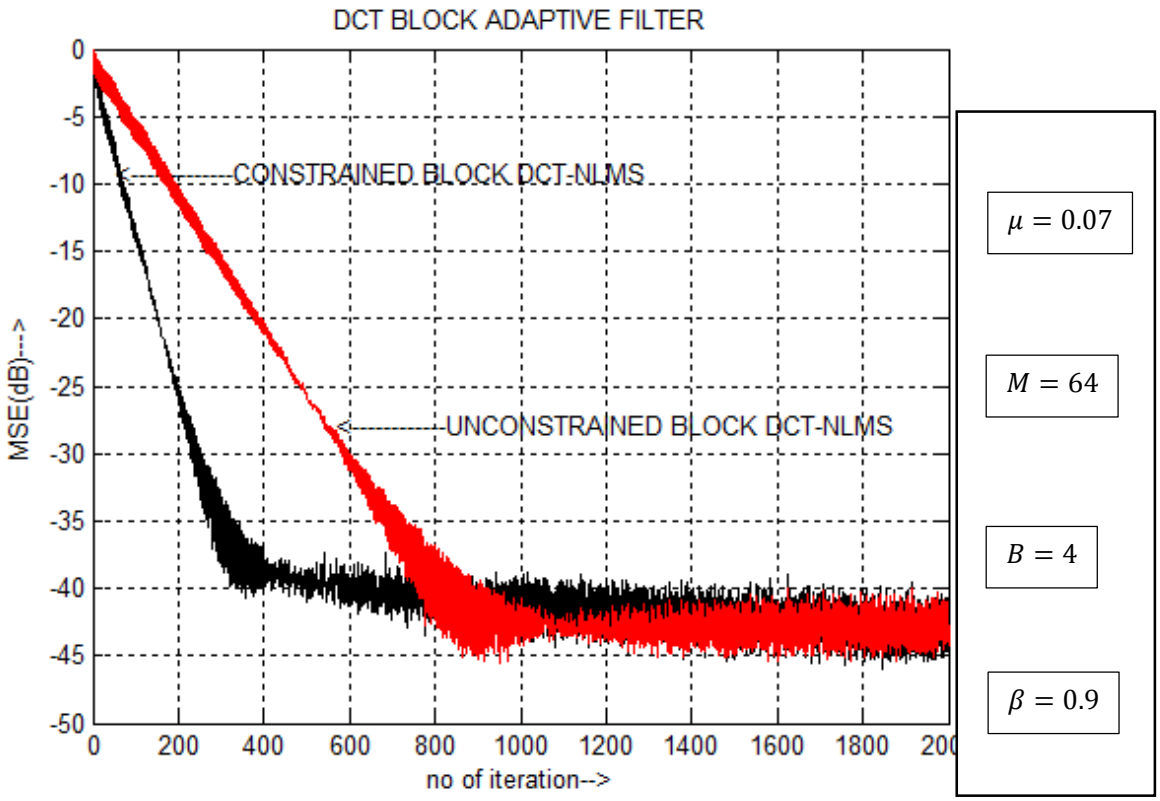


Figure-5. 6 DCT Block Adaptive Filter

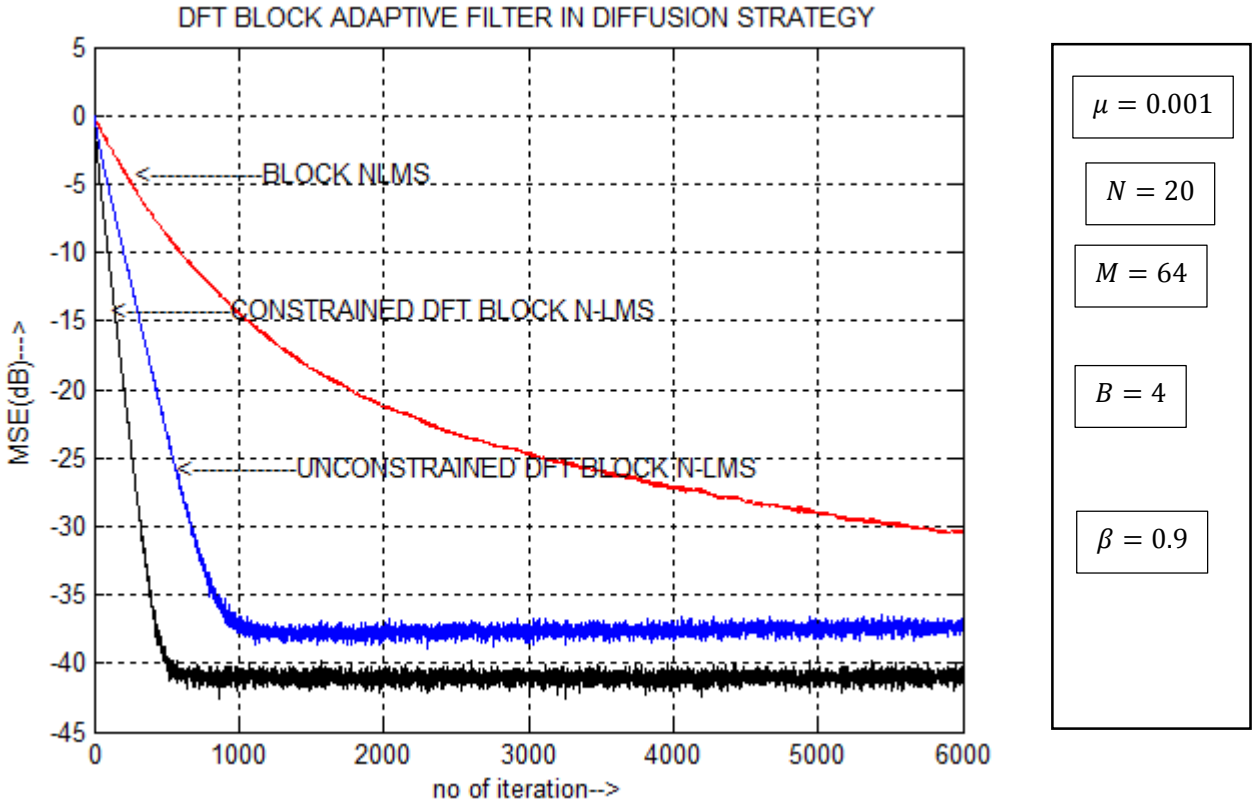


Figure-5. 7 DFT Block Adaptive Filter in Diffusion Strategy

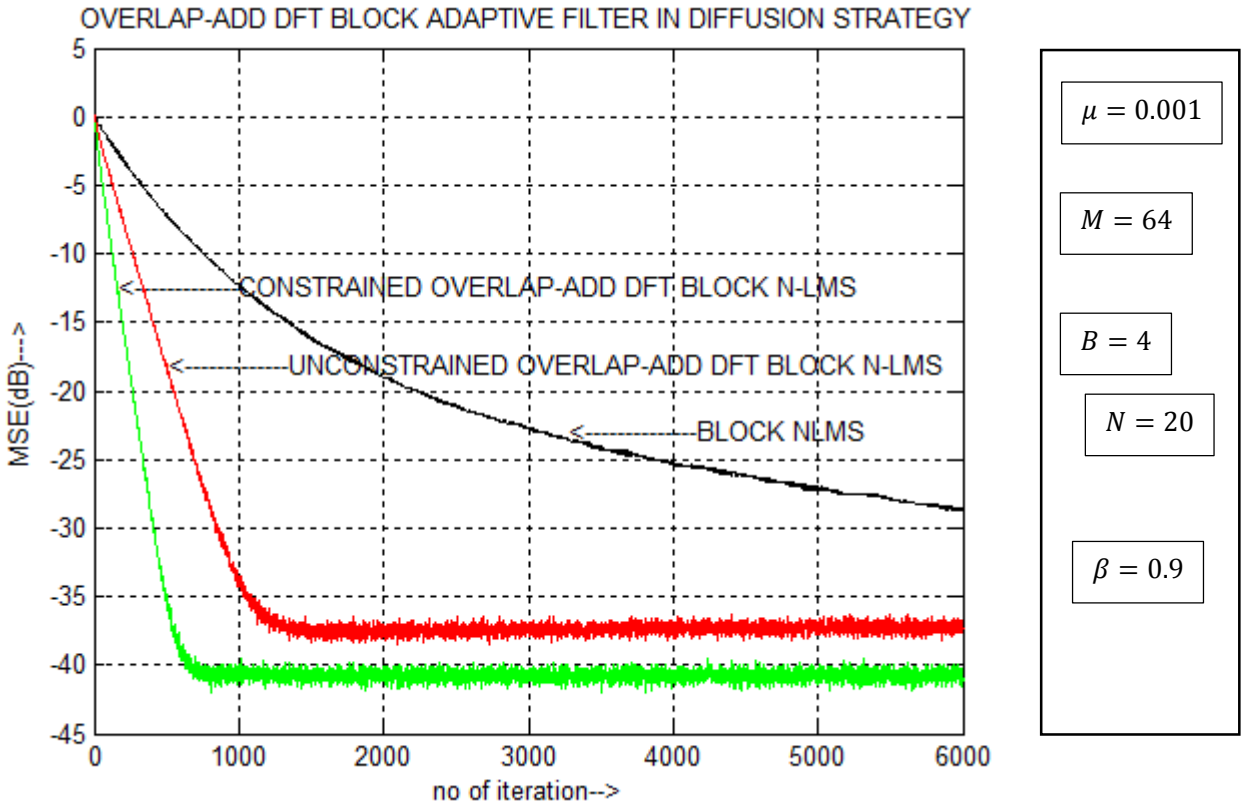


Figure-5. 8 Overlap-Add DFT Block Adaptive Filter in Diffusion Strategy

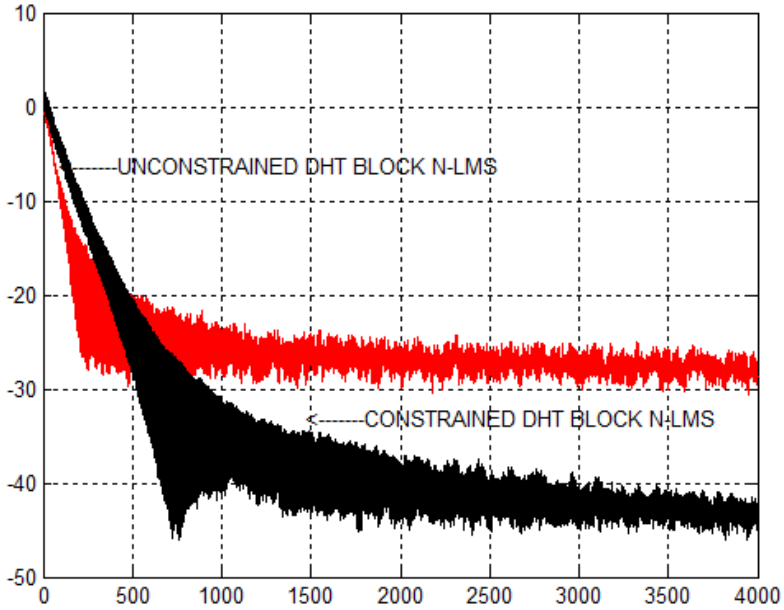


Figure-5. 9 DHT Block Adaptive Filter

Figure 1 DHT Block Adaptive Filter

CHAPTER 6
CONCLUSION & FUTURE WORK

By using unitary transforms DCT & DFT; the convergence performance is improved in both diffusion and incremental co-operation schemes. The DCT gives better results than DFT in convergence performance. The simulation results are shown above.

Similarly block adaptive filter of many types like DFT, DCT, DHT and Overlap-Add DFT are employed successfully in both constrained and unconstrained form. Constrained form of these filters gives better results. Mainly block filter are used to reduce the computational cost. Here we found that the cost is reduced to $O(M/B)$ operations per sample which is $O(M)$ operations per sample in simple filter. DFT and Overlap-Add block adaptive filters are giving better results and other two are more sensitive to noise. So DFT and Overlap-Add block adaptive filters are successfully employed in diffusion co-operation scheme and simulation results are shown above.

For future work point of view, we will study other transforms like KLT, DWT etc. and try to apply these transforms in block adaptive filters. The existing block adaptive filter expression should be look beyond the straight forward algebra to modify the existing filters.

Similarly in diffusion we will focus on other parameter to exchange with other nodes like input data, power of input in order to increase the efficiency of the distributed networks.

REFERENCE

[1] Kalouptsidis, Nicholas, and Sergios Theodoridis. *Adaptive system identification and signal processing algorithms*. Prentice-Hall, Inc., 1993.

[2] Sakai, Hideaki, Jun-Mei Yang, and Tetsuo Oka. "Exact convergence analysis of adaptive filter algorithms without the persistently exciting condition." *Signal Processing, IEEE Transactions on* 55.5 (2007): 2077-2083.

[3] Lopes, Cassio G., and Ali H. Sayed. "Incremental adaptive strategies over distributed networks." *Signal Processing, IEEE Transactions on* 55.8 (2007): 4064-4077.

[4] Sayed, Ali H., and Cassio G. Lopes. "Adaptive processing over distributed networks." *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 90.8 (2007): 1504-1510.

[5] Beaufays, Francoise. "Transform-domain adaptive filters: an analytical approach." *Signal Processing, IEEE Transactions on* 43.2 (1995): 422-431.

[6] Estrin, Deborah, et al. "Instrumenting the world with wireless sensor networks."*Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*. Vol. 4. IEEE, 2001.

[7] Rossi, Lorenzo A., Bhaskar Krishnamachari, and C-CJ Kuo. "Distributed parameter estimation for monitoring diffusion phenomena using physical models." *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*. IEEE, 2004.

[8] Li, Dan, et al. "Detection, classification, and tracking of targets." *Signal Processing Magazine, IEEE* 19.2 (2002): 17-29.

[9] Akyildiz, Ian F., et al. "A survey on sensor networks." *Communications magazine, IEEE* 40.8 (2002): 102-114.

[10] Lopes, CassioG, and AliH Sayed. "Diffusion least-mean squares over adaptive networks." *Proc. ICASSP*. Vol. 3. 2007.

[11] Sayed, Ali H., and Cassio G. Lopes. "Distributed recursive least-squares strategies over adaptive networks." *Signals, Systems and Computers, 2006. ACSSC'06. Fortieth Asilomar Conference on*. IEEE, 2006.

[12] Spanos, Demetri P., Reza Olfati-Saber, and Richard M. Murray. "Distributed sensor fusion using dynamic consensus." *IFAC World Congress*. 2005.

[13] Sayed, Ali H. *Fundamentals of adaptive filtering*. John Wiley & Sons, 2003.

[14] Shynk, John J. "Frequency-domain and multirate adaptive filtering." *IEEE Signal Processing Magazine* 9.1 (1992): 14-37.

[15] Moulines, Eric, O. Ait Amrane, and Yves Grenier. "The generalized multidelay adaptive filter: structure and convergence analysis." *Signal Processing, IEEE Transactions on* 43.1 (1995): 14-28.

[16] Tsitsiklis, John N., and Michael Athans. "Convergence and asymptotic agreement in distributed decision problems." *Automatic Control, IEEE Transactions on* 29.1 (1984): 42-50.

[17] Olfati-Saber, Reza, and Richard M. Murray. "Consensus problems in networks of agents with switching topology and time-delays." *Automatic Control, IEEE Transactions on* 49.9 (2004): 1520-1533.

[18] Li, Dan, et al. "Detection, classification, and tracking of targets." *Signal Processing Magazine, IEEE* 19.2 (2002): 17-29.
[19] Lopes, Cassio G., and Ali H. Sayed. "Distributed adaptive incremental strategies: formulation and performance analysis." *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. Vol. 3. IEEE, 2006.
[20] Xiao, Lin, Stephen Boyd, and Sanjay Lall. "A scheme for robust distributed sensor fusion based on average consensus." *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*. IEEE, 2005.